A Systems-Theoretic Framework for Safety-Driven Development of System Architectures

by

Justin Wei Siang Poh

B.S. Mechanical Engineering, Olin College of Engineering, 2016
S.M. Aeronautics and Astronautics, Massachusetts Institute of Technology, 2022

Submitted to the Department of Aeronautics and Astronautics
in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY IN AERONAUTICS AND ASTRONAUTICS
at the
MASSACHUSETTS INSTITUTE OF TECHNOLOGY

FEBRUARY 2025

Authored by: Justin Poh
Department of Aeronautics and Astronautics
December 23, 2024

Certified by: Nancy G. Leveson
J.C. Hunsaker Professor of Aeronautics and Astronautics
Thesis Supervisor

John S. Carroll
Gordon Kaufman Professor of Management, Post-Tenure

Natasha A. Neogi
Ph.D., Senior Technologist, Assured Intelligent Flight Systems, NASA

Accepted by: Jonathan P. How
R. C. Maclaurin Professor of Aeronautics and Astronautics
Chair, Graduate Program Committee

A Systems-Theoretic Framework for Safety-Driven Development of System Architectures

by

Justin Wei Siang Poh

Submitted to the Department of Aeronautics and Astronautics
On December 23, 2024 in Partial Fulfillment of the Requirements for the
Degree of Doctor of Philosophy in Aeronautics and Astronautics

Abstract

Modern complex systems are increasingly expected to exhibit emergent properties such as safety and security even as they become more complex, interconnected, and reliant on software than ever before. Because of this evolution in the characteristics of these systems, the methods available today for developing system architectures no longer provide systems engineers with adequate design support. As a result, it is becoming increasingly challenging for systems engineers to develop system architectures that exhibit emergent properties like safety.

This thesis addresses this problem by developing a safety-driven architecture development framework that enables the design of emergent properties such as safety into a system architecture from the beginning. The key idea is that the results from a hazard analysis process known as Systems Theoretic Process Analysis (STPA) should drive design decisions. The framework therefore starts with an initial STPA analysis of the system to determine how unsafe or undesirable behavior could occur. Structured and systematic processes are then provided to help systems engineers use the STPA results to develop the required control behavior of the system and explore possible system architecture options to implement that control behavior. This framework therefore enables systems engineers to make more informed early architectural design decisions driven by safety considerations. This framework is applied to an Urban Air Mobility (UAM) case study to demonstrate that it provides the necessary design support to enable the development and refinement of an air traffic management (ATM) architecture for UAM.

When creating a system architecture, assumptions may also need to be made to mitigate the inherent uncertainties and lack of detailed information about the system at that early stage of design. However, these assumptions are used as the basis for design decisions, and it is important that they remain valid to avoid flaws in the architecture arising when underlying assumptions become invalid. Thus, this thesis also develops and demonstrates a supporting framework to help identify these underlying assumptions and ensure they remain valid both during system development and after the system is placed into operation.

Thesis Supervisor: Nancy G. Leveson, Ph.D.
Title: J.C. Hunsaker Professor of Aeronautics and Astronautics

Acknowledgements

I started this journey toward earning my doctorate 4.5 years ago, having spent the prior 4 years working on the development of self-driving vehicles. When I first decided to pursue a graduate education, my goal was to improve my abilities as a systems engineer by learning more about systems engineering and system safety. I have accomplished so much more than that since then and this dissertation would not have been possible without the support and mentorship of numerous people along the way.

First, I want to express my sincere gratitude to Professor Nancy Leveson for accepting me into her research group and not only teaching me about system safety but also showing me a whole new perspective on systems engineering. Her views on systems theory and how we should be designing safety into systems have strongly influenced my work. Thanks to her mentorship, I am a better systems engineer now because I better understand how to design safety and other properties into systems. In addition, I will forever be grateful for the time she spent giving me feedback on my writing and presentations and helping me to refine my communication skills.

I am also deeply grateful to the members of my Ph.D. committee for all their guidance and feedback. Dr. Natasha Neogi spent many hours helping me to both fine-tune the application of my ideas and to contextualize my work within the existing air traffic management research. I am extremely grateful for all her incredible insights and her ability to see the significance of my work, sometimes before I could. Professor John Carroll also frequently reminded me not to forget the real-world context in which systems are designed, much of which is often abstracted away or simplified in traditional systems engineering practices. Together, my committee played a pivotal role in guiding and shaping my research, and I will forever be grateful for their guidance.

I also owe many thanks to my readers who have provided invaluable feedback and input on my work since my thesis proposal. The many discussions I have had with Dr. Bill Young about various security and systems engineering topics have helped me see the contribution of my work more clearly. Dr. John Thomas has a wealth of experience with STPA and invaluable teaching skills and I am grateful for the many hours he spent discussing my research with me and helping me to refine my dissertation to better communicate those contributions. To both of my readers, thank you so much for all the discussions we've had and the feedback you've given me.

Next, I want to thank my research group for not only their friendship but also the help they've given me throughout my time at MIT. Dr. Andrew Kopeikin and I worked together on research early on and the depth of his experience as an aviator and aerospace engineer helped me better understand the world of aviation and air traffic management, especially since I was just starting out in this field at that time. To Brittany Bishop, Rodrigo Rose, Polly Harrington, Alex Hillman, Lauren Guttierez, Natalie Basnight and Braden Brower, thank you for all the research discussions and casual conversations we've had over the years. It has been a pleasure to learn from you all.

Finally, to my wife Sophia, words cannot express the gratitude I feel for the love and support you have given me and our family (of 2 dogs) throughout this journey. I could not have completed this dissertation without you and I will be forever thankful for you.

Table of Contents

List of Figures

List of Tables

Chapter 1 Introduction

1.1 Motivation

Developing complex systems today is becoming more challenging than ever before because of both an evolution in the characteristics of these systems and a greater emphasis on the properties that these systems must exhibit [1]. On the one hand, the demand for greater functionality and greater productivity has given rise to systems with more components that are more interconnected and interdependent than systems of the past [2]. In addition, there is an increasing desire to use automation and software to augment or enhance the capabilities of these systems. Examples of this include the development of self-driving vehicles by the automotive industry [3], the wide variety of automation and autonomous functions being introduced onboard current and future aircraft [4, 5] and the development of Terrain Relative Navigation that was used to guide the landing of NASA's Mars rover Perseverance in 2021 [6].

On the other hand, these systems are also increasingly expected to exhibit properties such as safety, security, and sustainability. For example, in the automotive industry, there is much greater emphasis on functional safety and cybersecurity today as a result of the potential safety concerns associated with self-driving vehicles operating on public roads [3]. Similarly, in the space industry, the increasing number of spacecraft operated by government and commercial organizations has resulted in increased attention being paid to reducing debris and pollution in space for the safety and sustainability of future space missions [7].

Properties like these are known as Emergent Properties because they are the result of (i.e., they *emerge* from) the behavior of the system and the interactions between system components [2, 8]. A system will only exhibit these properties if it is designed to achieve the required behavior while avoiding undesirable behavior and interactions between components.

One set of early design decisions that have an important influence on the system's behavior are those made to create a system's architecture [9]. A *System Architecture* is defined as an abstract description of the entities of a system and the relationships between those entities [9]. It is therefore important to develop an appropriate system architecture to ensure that the desired emergent properties like safety are exhibited by a system. Unfortunately, current methods for developing system architectures do not provide sufficient support for designing a system architecture to achieve emergent properties like safety.

The goal of this dissertation is to address this problem by developing an alternative approach to architecture development that is more suitable for architecting modern complex systems and that helps systems engineers to be more successful in designing emergent properties into their system architectures from the beginning.

1.2 Challenges in Architecting Complex Systems

To design a system architecture to achieve emergent properties such as safety and security, systems engineers need to be able to identify the requirements necessary to achieve those properties. They can then determine the behavior and structure of the system needed to meet those requirements [2]. Unfortunately, as systems have become more complex and software-intensive, current methods for developing system architectures have become limited in their ability to help systems engineers make appropriate design decisions.

One limitation of current methods is that they are not focused enough on the safety-relevant interactions in a system. Many current methods focus too much on the physical components and interfaces of a system. The important safety-related interactions thus become hidden amongst the myriad of system components and interfaces. As a result, it can be difficult for systems engineers to fully comprehend how their design decisions will impact the behavior of the system, and they may inadvertently introduce flaws in the system architecture as they create it.

Another limitation of current methods is that they lack a systematic process for identifying potential architecture options. Although these methods suggest general design principles to use such as maximizing modularity or minimizing coupling [10, 11, 12], they provide minimal guidance on when to apply which design principles. Systems engineers are thus left to make these decisions using their experience and engineering judgement.

Finally, the last limitation of current methods is their reliance on quantitative metrics for evaluating and comparing architecture options. Historically, detailed system architectures have been compared using quantitative physical metrics such as mass or thermal performance [13]. However, for properties like safety, it is much harder to identify similar types of measurable quantitative metrics, especially during the early stages of system development.

1.3 A Systems-Theoretic Approach to Architecture Development

To address these limitations, a new approach to architecture development is needed that provides more support to help systems engineers design emergent properties like safety into a system architecture from the beginning [2, 14]. This research proposes Systems Theory as the foundation for this new approach.

Systems Theory is uniquely suitable because it recognizes the importance of considering the system as a whole instead of just focusing on the individual components. In addition, a key idea in Systems Theory is that emergent properties are realized when sufficient constraints are enforced on the interactions between components. When these constraints are adequately enforced, the necessary interactions occur while undesirable interactions are avoided [2, 8].

This is an important characteristic of emergent properties because it suggests that to achieve them, the system must include sufficient control over the behavior of the system components and the interactions between them to prevent undesirable behavior. For example, to ensure the safe operation of a self-driving vehicle, there must be adequate control to ensure the vehicle navigates safely on public roads without colliding with objects or other road users while getting passengers or cargo to the correct destinations at the desired time. Because Systems Theory focuses on control in a system, it provides a useful theoretical foundation for a new approach to architecture development that provides more appropriate design support to help systems engineers design emergent properties like safety into a system architecture.

In [15], an initial version of a systems-theoretic approach to architecture development was developed. A key strength of the approach developed in [15] was that it provided guidance for using the results from a hazard analysis method called Systems Theoretic Process Analysis (STPA) to identify system requirements and the required system-level behavior. However, a major limitation was that it lacked a similarly structured and systematic process for creating architecture options and comparing them. As such, this research aims to extend the initial systems-theoretic approach developed in [15] and address this key limitation.

1.4 Research Objective and Contributions

The goal of this dissertation is to improve the ability of systems engineers to design emergent properties like safety into a system architecture by extending the initial systems-theoretic approach developed in [15]. The objective of this research can therefore be stated as follows.

Research Objective: To create an architecture development framework that provides structured and systematic processes for creating and assessing system architectures.

To address the challenges described in Section 1.2 and achieve the research objective, there are three main contributions of this research.

Contribution 1: *A structured process for comparing identified architecture options based on safety-relevant criteria to support the development of a preferred architecture option*

This contribution addresses the reliance of current methods on quantitative metrics for evaluating and comparing architecture options. Especially during the early stages of design, when many design details about the system are not yet known, it can be challenging to identify appropriate quantitative metrics for properties such as safety and evaluate the performance of an architecture with respect to those quantitative metrics.

Instead of relying on quantitative comparisons of system architectures, this research develops a structured process for performing a qualitative, control-oriented comparison of architecture options. By analyzing each architecture option under consideration using STPA, the identified scenarios can be compared to determine the benefits and tradeoffs of each architecture option. Ultimately, these benefits and tradeoffs can be used to inform a decision about the preferred architecture that best achieves the desired emergent properties.

Contribution 2*: More structured and systematic processes for developing and refining the system behavior and architecture necessary for safety and other emergent properties to be achieved*

This contribution addresses the lack of appropriate design support provided by current methods. This lack of support is the result of current methods not focusing enough on the control-oriented interactions in a system and only suggesting general design heuristics to help systems engineers create a system architecture. To address these limitations, the architecture development framework developed in this research focuses first on defining the control behavior that a system must achieve to successfully exhibit properties such as safety. Only after the required behavior is defined is a system architecture created to implement it.

To create the required control behavior, this research extends the guidance provided in [15] to create a more structured process for identifying the required control elements. The process also defines how to iterate on the behavioral design. This iteration allows systems engineers to both fix any flaws and more thoroughly explore the behavioral design space for a system.

Once the required behavior has been defined, this research develops a more structured and systematic process for using STPA to iteratively identify architecture options that are worth evaluating and comparing. Using insights gained from the analysis of these options, the process helps systems engineers to incrementally develop and refine the system architecture.

Contribution 3: *A supporting framework for identifying and accounting for assumptions made during architecture development*

Especially when designing new systems that do not yet exist, early design decisions are made under significant uncertainty. For example, the environment in which that system will operate may not be fully known. To make design decisions despite these uncertainties, assumptions are typically made about what the system's behavior or operating environment might be in the future [16]. However, the effectiveness of those design decisions becomes contingent on the underlying assumptions remaining valid. If an assumption becomes invalid, the effectiveness of the associated design decisions may be compromised, and flaws may arise in the system design.

For this reason, although this research is primarily focused on how to make appropriate architectural design decisions, a supporting framework for identifying and accounting for underlying assumptions was also developed and demonstrated.

1.5 Case Study: Air Traffic Management and Urban Air Mobility (UAM)

To demonstrate and evaluate the architecture development framework created in this research, the framework is applied to develop an air traffic management (ATM) architecture for the National Airspace System (NAS) to enable the implementation of Urban Air Mobility (UAM). This section provides a brief overview of the architecture problem to be solved.

UAM is a relatively new mobility concept that envisions using small aircraft to transport passengers or cargo on demand within an urban area [17, 18], similar to the ride hailing services provided by Lyft and Uber today. To realize this novel transportation concept, one of the major challenges of interest in this research is the integration of UAM into the NAS.

There is broad recognition that the centralized approach used to manage air traffic in the NAS today will not feasibly accommodate the addition of UAM flights [18, 19, 20, 21, 22, 23] because they have very different characteristics than today's air traffic. For example, UAM is expected to operate at a higher traffic density, faster pace of flight operations, and perform flights more on-demand compared to today's regularly scheduled commercial air traffic [18, 19, 23]. In addition, UAM aircraft are anticipated to spend most of their flight time flying low and slow over densely populated urban areas [18, 23] unlike today's air traffic that spend most of their flight time at high altitudes away from densely populated areas.

The existing ATM architecture was not built to manage air traffic with these characteristics. From a workload perspective, the current ATM architecture depends on human air traffic controllers having centralized control over air traffic [24] and they would be overwhelmed by the increased traffic density and pace of UAM flight operations [19]. In addition, the current ATM architecture relies on having enough time and extra space to respond to unexpected incidents such as emergencies or weather disruptions. However, with the increased traffic densities and the low and slow flight of UAM aircraft over populated areas, there will be much less time and space available to respond to disruptions or emergencies when they arise.

For these reasons, it is necessary to re-design the ATM system architecture to accommodate UAM flights without compromising the level of safety of the NAS. The goal of this case study is therefore to develop an appropriate ATM system architecture that will be able to safely manage UAM flights alongside existing aviation operations.

Because the ATM system is extremely complex, it needs to be designed incrementally to ensure that the development effort remains intellectually manageable. This dissertation therefore develops the ATM architecture for UAM over two design iterations to demonstrate how this architecture development framework can be used to develop and refine a system architecture based on incremental insights about the architecture gained during prior iterations.

1.6 Hypotheses and Evaluation

This research explores two main hypotheses, and their evaluation provides support for the first two research contributions described in Section 1.4. While Contribution 3 is demonstrated as part of the UAM case study, no formal evaluation is performed.

Hypothesis 1: *A systems-theoretic approach can identify relevant criteria for comparing architectures and evaluating their ability to achieve emergent properties*

This hypothesis is evaluated in Chapter 4, where the first design iteration of the UAM case study is performed to create an initial collision avoidance ATM architecture for UAM. As part of this design iteration, two architecture options are compared: (1) a centralized collision avoidance architecture and (2) a decentralized collision avoidance architecture. The benefits and tradeoffs identified for these two architecture options using this architecture development framework are then compared to the benefits and tradeoffs identified in similar comparisons that have been performed in the existing literature. This evaluation demonstrates that this framework identifies not only the benefits and tradeoffs that have been found in the existing literature but also additional ones that provide a more comprehensive understanding of the various ways that an architecture option is able or unable to achieve safety.

Hypothesis 2: *A systems-theoretic approach can support making informed design decisions to iteratively develop and refine the architecture for a system*

This hypothesis is evaluated in Chapter 5 by completing a second design iteration for the UAM case study to refine the initial collision avoidance architecture created in design iteration 1. After both design iterations are complete, the progression of the ATM architecture over the two iterations is evaluated. This evaluation demonstrates that the framework enables incremental refinement of a system architecture and provides the necessary support to help systems engineers make informed design decisions as they make these refinements.

1.7 Scope

This research is scoped in several ways to ensure appropriate depth of focus. First, the architecture development framework created in this research is intended to be used during the concept and architecture development phase of the systems engineering V-model [25, 26]. The framework begins after stakeholder analysis is complete and assumes that a prioritized set of stakeholder needs and a statement of the system's purpose is already available. The framework ends with the selection of a system architecture that is intended for use in downstream detailed system design and verification and validation activities. Thus, this research will not consider the process of eliciting and prioritizing stakeholder needs or the process of verifying and validating the system. This research will also not consider the creation of a detailed system design. This ensures that the focus of this research remains on early-stage system architecture development.

Second, this research focuses on the *process* of developing a system architecture and does not consider the tools and methods for documenting it. This specific focus ensures that a clear process is defined for creating system architectures using this new approach before considering the methods and tools needed to support that process. Thus, architecture description frameworks such as the Department of Defense Architecture Framework (DoDAF) [27] or The Open Group Architecture Framework (TOGAF) [28] will not be considered. Similarly, the use of specific modeling languages such as the Systems Modeling Language (SysML) [29] are also not included within the scope of this research.

Finally, the framework created in this research focuses primarily on designing safety into system architectures. In addition to preventing traditional losses such as loss of life, injury, or damage to property, it also includes broader notions of safety such as loss of mission. Although the author believes this architecture development framework could be applied to design other emergent properties into systems besides safety, this research focuses primarily on safety.

1.8 Organization of Dissertation

The remainder of this dissertation is organized as follows.

Chapter 2 reviews the available literature from several engineering disciplines to identify the different types of approaches that are currently used to develop system architectures. The limitations of these approaches are discussed to identify specific gaps that this research needs to address. Then, an overview of systems theory and STPA is provided to justify their use as the foundation for the architecture development framework created in this research.

Chapter 3 describes the development of the safety-driven architecture development framework. First, the overall approach to architecture development is described by applying key concepts from systems theory. Then, an overview of the safety-driven architecture development framework is provided followed by a description of the processes contained within it.

This architecture development framework is then applied over two design iterations to develop an ATM architecture for the NAS that can manage UAM air traffic alongside existing air traffic. The first design iteration focuses on developing a high-level collision avoidance architecture for the NAS to show how the safety-driven architecture development framework can be used to generate a system architecture based on hazard analysis results. Chapter 4 presents the results from this first design iteration.

The second design iteration then focuses on refining the selected high-level collision avoidance architecture in iteration 1 to obtain a more detailed definition of the collision avoidance architecture for the NAS. This design iteration shows how the safety-driven architecture development framework can also be used to incrementally refine a system architecture. The results from this second design iteration are presented in Chapter 5.

Chapter 6 develops and demonstrates the supporting framework that was developed to help identify underlying assumptions during the architecture development process and account for them as the system architecture is developed.

Finally, Chapter 7 summarizes the conclusions of this dissertation, discusses some of its limitations and describes possible directions for future work.

Chapter 2 Literature Review

Although the process of developing a system architecture is typically considered to be a systems engineering activity, Systems Engineering is not the only discipline that has created methods and approaches for developing architectures. In the late 1960s and 1970s, as the size and complexity of software systems began growing, there was recognition that methods were needed to facilitate the creation of good software architectures [30, 31]. Later, as systems engineers began to contend with similar increases in size and complexity of more general engineered systems, many of the ideas for architecting software systems were adapted by the Systems Engineering community to design these engineered systems. More recently, the field of Product Design, which is closely related to Systems Engineering, also developed methods for creating good product architectures [32].

As a result of these past research efforts, a wide variety of different methods for architecture development already exist. Instead of simply coming up with yet another new architecture development method in this research, it is important to evaluate these existing methods to better understand the limitations that make them ill-suited for designing modern complex systems that have the characteristics described in Chapter 1. These limitations can then be used to inform the development of the new framework.

2.1 Current Methods for Architecture Development

In this research, four main types of approaches were identified in the disciplines identified in the previous section: (1) Decomposition-based methods, (2) Reuse-based methods, (3) Quantitative methods, and (4) Flow-based methods.

2.1.1 Decomposition-Based Methods

One of the most commonly used approaches for developing system architectures is decomposition. Decomposition, also known as analytic reduction, is the process of dividing up a system into its constituent parts or functions [10]. Not only do decomposition-based methods exist in all of the disciplines surveyed in this research, but decomposition is also the foundation for processes recommended in safety standards such as ISO 26262 [33] and ISO 21448 [34].

For example, in software engineering, one focus in the 1970s was on modularity and identifying effective ways to divide a software program up into modules [35, 36, 37]. Two main ideas were proposed. The first is information hiding where the goal is to divide a system up into modules such that design decisions contained within one module are hidden from the rest [36]. The second is stepwise refinement where the goal is to incrementally divide a software program into a series of subtasks, gradually making more detailed design decisions [37].

In Systems Engineering and product design, decomposition is also commonly used to identify the system requirements and functions [26, 32, 38, 39] and one popular group of methods for doing this is Model-Based Systems Engineering (MBSE) methodologies. In a widely cited 2008 paper [40], Estefan reviews six of the most popular MBSE methodologies including INCOSE's Object-Oriented Systems Engineering Methodology (OOSEM) [11] and others [12, 41, 42, 43, 44]. Although each method contains slight variations, they all follow an overall process that is like that shown in Figure 1 to generate system requirements, the necessary functions, and the system architecture to implement those functions.

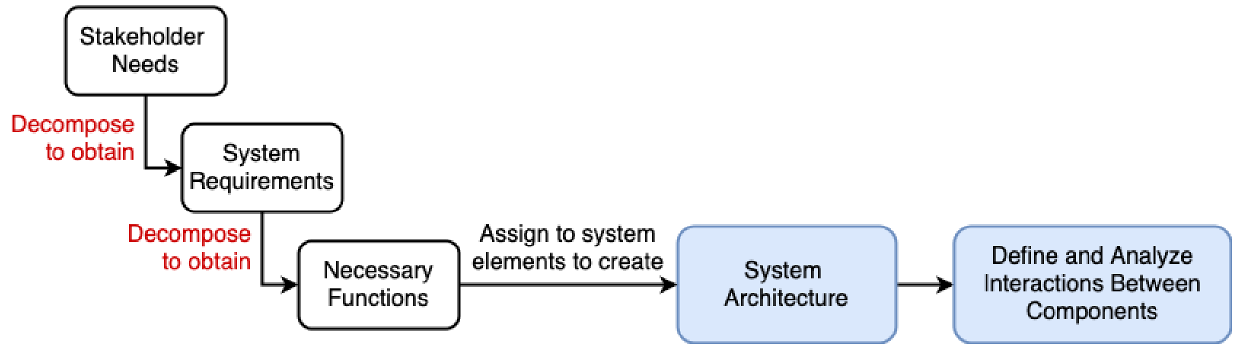


Figure 1: Today's decomposition-based architecture development approach

The result of this decomposition-based process is a system architecture that is typically represented using an object-oriented model that also focuses primarily on the components (i.e., objects) and the interfaces between them. As an example, Figure 2 shows the block diagram for a satellite drawn using an object-oriented modeling language called Systems Modeling Language (SysML) [29]. In Figure 2, the satellite is represented in terms of the objects or components (e.g., electrical power subsystem) and the interfaces between them (e.g., data flows, power cables).

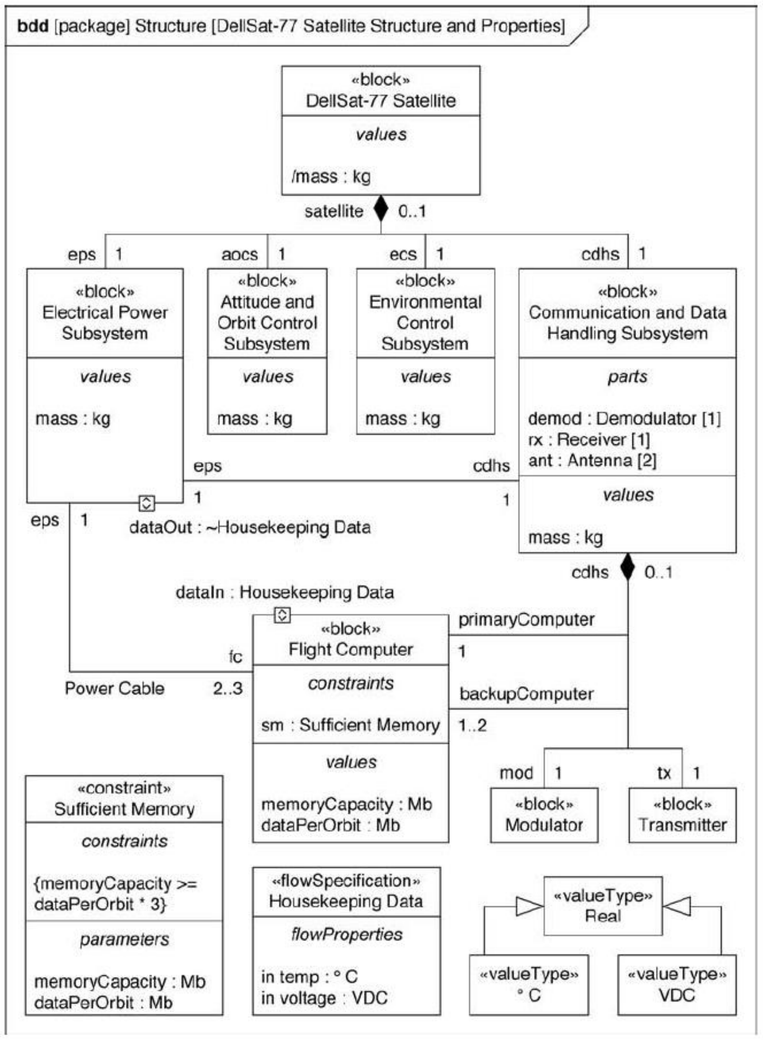


Figure 2: Example system block diagram for a satellite drawn in SysML (from [29])

Limitations of Decomposition-Based Methods

These approaches could be used to design simpler systems of the past because those systems contained components that behaved relatively independently of each other. As a result, it was possible to assume that a system could be divided up into components before identifying and analyzing the interactions. It was also assumed that the system design will be successful if the behavior of each system element is well understood and has clearly defined interfaces. Unfortunately, these assumptions are not necessarily valid for today's software-intensive complex systems [2, 9, 45]. Instead, the design process needs to be more focused on the control-oriented interactions in a system. As discussed in Chapter 1, it is these control-oriented interactions that determine if the system can adequately enforce the safety constraints and thus exhibit the desired emergent properties.

One of the key limitations of decomposition-based approaches is therefore that the methods and the underlying system models they use focus too heavily on the components and interfaces in a system. The control-oriented aspects of the system thus become obscured or hidden, making it more difficult for systems engineers to recognize the critical interactions that define the control behavior of the system. For example, the block diagram of the satellite in Figure 2 shows numerous physical interactions between the satellite's subsystems but not the interactions that ensure the satellite is at the right orbital altitude and in the right orientation to fulfill a mission.

Another key limitation is their lack of guidance for how to create the system architecture or identify possible architecture options. Because there are typically multiple ways to divide up a system into components [26, 39], rules of thumb known as partitioning heuristics are sometimes used to help systems engineers decide how to divide up the system into modules. For example, Design Structure Matrices (DSMs) [46] help identify ways to modularize a system that minimizes the connections between modules and maximizes a module's independence. Some additional examples of partitioning heuristics suggested in program design and MBSE methodologies include:

1. Maximizing cohesiveness [11, 47]
2. Information hiding [11, 35]
3. Enabling component reuse [11, 47]
4. Minimize difficulty in making changes [11]

Unfortunately, these heuristics do not always lead to a good system architecture for any system. A systems engineer must therefore know when to apply which heuristics [48]. However, decomposition-based methods typically offer little guidance and include minimal system-level analyses to help systems engineers choose the right heuristic(s) for a given system. For example, many of the MBSE methodologies use system use cases to inform the decomposition of a system into functions [11, 42, 47].

In summary, decomposition-based methods are limited in their ability to support the development of system architectures for modern complex systems for two reasons. First, the underlying object-oriented system models do not adequately emphasize the control-oriented aspects of a system. Second, they lack a systematic process for creating the system architecture or identifying possible architecture options.

2.1.2 Reuse-Based Methods

Another approach to architecture development that has received significant research attention in systems engineering and software engineering is reuse-based methods. Reuse-based methods are an approach to architecture development that identifies fundamental aspects of a system design that can be reused (like a template) to solve commonly occurring design problems. The goal of these methods is to help software engineers or systems engineers to more quickly identify useful architectures by leveraging the lessons learned from past design efforts.

In the mid to late 1990s, design patterns [49] or architectural styles [30] were created to help capture and share design experience about how to structure a software system. In essence, they defined a configurable or re-usable system model [49, 50] that could be tailored to suit a specific system being designed and they have sometimes helped to identify less obvious system structures that a software developer might not have identified on their own [49].

Another reuse-based method in software engineering that is similar to design patterns is software design frameworks. Software design frameworks describe a commonly occurring problem and the required objects and system structure needed to solve that problem [51]. In addition, they also contain the code for a reusable main program and the developer decides what components to plug into it [51, 52]. A commonly cited example of a software design framework is the Model/View/Controller (MVC) framework for implementing a graphical user interface [51].

The concept of reusable architectures and patterns has also been applied in Systems Engineering for creating system architectures. For example, INCOSE and NASA both recommend using reference architectures when creating the system architecture [26, 39]. Like patterns, a reference architecture is a system architecture for a previous system that a systems engineer can use to help decide how to architect a new system.

INCOSE has also explored the use of pattern-based design methods in systems engineering and their MBSE patterns working group [50] created Pattern-Based Systems Engineering (PBSE). In PBSE, a pattern is a reusable or configurable system model. PBSE is thus an extension of traditional MBSE methodologies with additional methods for managing patterns and configuring them to suit the needs of a particular project [50, 53].

Another reuse-based method that has been proposed in systems engineering is Platforming [54]. Platforming is defined as the sharing of components or processes across a family of products with the goal to reduce the development costs and lead times by sharing these costs across multiple products [54]. For example, the automotive industry uses platforming to create families of vehicles that all use a common foundational vehicle platform [54].

Limitations of Reuse-Based Methods

Although there are potential benefits to using reuse-based methods to create system architectures, there are also important limitations to consider. First, although the use of reuse-based methods is often motivated by the ability to share design knowledge and speed up the design process, these benefits have not been empirically validated. In a 2012 paper [55], Zhang and Budgen reviewed the literature on software design patterns up to 2009. Although they found good support for the claim that using patterns improves communication between software developers and maintainers, they found no evidence that patterns are effective at helping novices learn about design by sharing design knowledge. They also found inconclusive evidence that the use of patterns has any impact on the productivity of developers or the quality of the software they produce. In fact, [56] notes that reuse has resulted in significant accidents in aerospace systems in the past.

Another limitation when using patterns or reference architectures is that it can be challenging for systems engineers and software developers to make decisions about which patterns to use in their design. This is because studies have found that focusing on reuse may not necessarily be the best approach for all systems [55] and choosing which patterns to apply still requires some prior design experience. Zhang and Budgen highlight this problem in [55] when they note that software developers first need to gain experience by seeing how others applied a pattern before they can know how and when to use that pattern in their own designs [55].

Finally, the use of patterns or reference architectures assumes that a good architecture has already been identified for a given problem. For new versions of existing systems or new systems that bear significant similarities to existing systems, patterns or reference architectures may be useful to avoid having to start a design from scratch. However, for new systems that have never been built before (such as UAM), a "good" architecture will not be available. This makes patterns and reference architectures of limited use when designing new types of systems. Furthermore, attempting to use a pattern and reference architecture could limit the opportunities for systems engineers to come up with new architectures that might perform better for their specific system than the pattern or reference architecture.

In summary, the main limitations of reuse-based methods are that many of the claimed benefits have not been validated and these methods can be challenging to apply because they provide little guidance on how to select the best pattern or reference architecture for a given system. In addition, although these methods may be beneficial for systems that have established some amount of design precedent, they are limited in their ability to help systems engineers design new types of systems where a good architecture has not yet been identified.

2.1.3 Quantitative Methods

In the decomposition-based and reuse-based methods reviewed above, the focus is primarily on choosing the best way to decompose a system into modules and it is this choice that drives the creation of a system architecture. However, these methods typically only consider a few possible architecture options.

An alternative approach is to perform tradespace exploration. A tradespace is defined as the set of architectures represented on a space defined by two or more metrics [10]. Quantitative methods aim to explore the architectural tradespace for a system more thoroughly by systematically identifying possible architecture options and then quantifying and comparing the performance of these options with respect to a set of performance criteria. This quantitative comparison allows the system designer to make a data-driven decision about which option is the best. Thus, in quantitative methods, the performance criteria drive the creation (and selection) of a system architecture. There are three main types of quantitative methods: (1) enumeration and evaluation methods, (2) optimization methods, and (3) simulation methods.

Enumeration & Evaluation methods

The main goal of enumeration and evaluation methods is to first identify all feasible architecture options and then evaluate and compare the performance of each one. One simple way to enumerate architecture options is to use morphological matrices [10]. A morphological matrix is essentially a table where each row represents a design variable, and each column represents an alternative value (option) for that design variable. Architecture options are created by selecting one value for each design variable. For more complex system architectures, an alternative approach is to represent the design variables and the connections and constraints between them using a graph [57, 58, 59]. The graph is then traversed and values for each design variable are chosen to create architecture options.

Once the architecture options are enumerated, the performance of each architecture option must then be evaluated with respect to a set of performance metrics. One common evaluation approach is to use designer-provided equations that specify the relationship between architectural features and the selected performance metrics [58, 59]. Alternatively, a method called Value Assessment of System Architectures Using Rules (VASSAR) proposed to evaluate architecture options based on the extent to which it satisfies the stakeholder requirements [57].

Once the performance of all architecture options has been evaluated, they can be compared to identify tradeoffs between them. This comparison is sometimes done graphically using pareto front plots [10, 57, 58, 59]. A pareto front plot is essentially a scatter plot where each point represents an architecture option and the axes represent the performance metrics (e.g., total utility and cost) that the architecture options are evaluated against. For example, Figure 3 shows a pareto front plot generated from a retrospective analysis of different mission architectures developed in the 1960s for the Apollo program [59]. Each point represents a mission architecture option that is plotted based on its mission success probability (a measure of utility) and initial mass to low earth orbit (IMLEO) (a measure of cost).

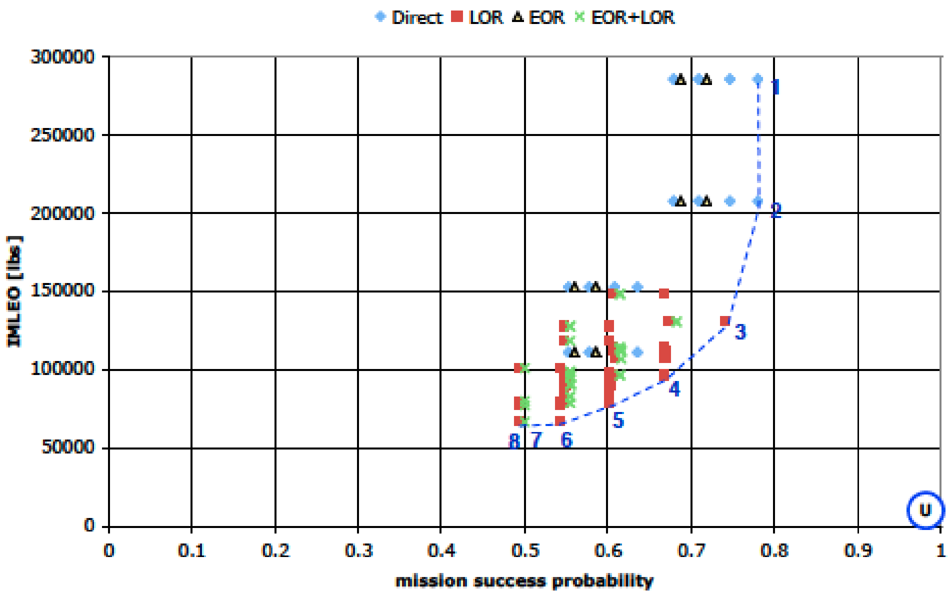


Figure 3: Pareto plot for possible apollo mission architectures (reproduced from [59])

The pareto front plot thus helps a systems engineer to visualize the tradespace and quickly identify the architecture options with the best performance. It can also help to identify tradeoffs between architecture options that have similar overall performance but exhibit better performance on one metric at the cost of worse performance on another metric.

Optimization Methods

While evaluation and enumeration methods provide good coverage of the tradespace for a system, they can require a significant amount of time and computational resources to evaluate the large number of architecture options that are generated. To reduce these costs, optimization methods identify the best (optimal) design for a system by iteratively searching the architectural tradespace. At each iteration, the goal is to identify architecture options that offer incrementally better performance than the architecture options found in previous iterations. Iteration stops when the architecture option with the best performance has been found.

One example of an optimization method used in mechanical engineering and product design is Axiomatic Design [60], a mathematical approach to system design where a system designer must select the design parameters that define the system. Matrix-based design equations then define how the design parameters satisfy the functional requirements [60].

Using these matrix-based design equations, Axiomatic Design provides two design axioms to help a designer choose the best design parameters for their system. The *Independence Axiom* focuses on maximizing the independence between the functions of the system. Based on this principle, a system designer should select design parameters that minimize the coupling between the functional requirements. The *Information Axiom*, then, provides a quantitative approach for determining how good the design is by calculating the probability that a system successfully achieves its desired performance with respect to the functional requirements.

While axiomatic design has been applied successfully to the design of physical systems, it is more difficult to apply to other types of system design because of its narrow focus on functional independence and information content. In addition, axiomatic design tends to be focused on design within a specific discipline whereas systems engineers need to consider the concerns from multiple disciplines simultaneously when selecting an optimum design.

To address this need to optimize across engineering disciplines concurrently, Multi-Disciplinary Design Optimization (MDO) methods were created. Instead of making discipline-specific design decisions sequentially (e.g., structural design, then thermal design), MDO methods optimize the system design over the constraints of multiple disciplines simultaneously. These methods were initially used to design aerospace systems where strong coupling between engineering disciplines made it challenging to design systems one discipline at a time [61].

In MDO, the design problem is formulated mathematically using equations that define (1) the design variables and the range of possible values, (2) the objective function used to calculate the performance of an architecture option based on the values of the design variables, and (3) the constraint functions used to quantify the constraints that an architecture option must meet to be considered feasible [13].

Once the problem has been formulated, a wide variety of different optimization algorithms can be used to identify the optimal values for the design variables. [62] provides a good overview of the different classes of optimization algorithms and how to select which algorithm to use based on the characteristics of the objective function and constraint functions. Although many of the optimization algorithms used by MDO methods are designed to solve for continuous design variables such as length or mass, there are also optimization methods that are designed specifically for optimization problems involving discrete design variables [63, 64].

Simulation Studies

Finally, a third type of quantitative method for analyzing and evaluating system architectures is simulation studies. Although simulation studies do not help to create architecture options, they provide a way to evaluate and compare them by “operating” a system virtually. Systems engineers can then obtain quantitative data about the aspects of a system’s behavior that are of interest before any software or hardware is developed. Simulation studies can thus be used to compare different architecture options operated in the same simulated environment or scenario.

Simulations are becoming an increasingly popular tool for evaluating the behavior or performance of a wide variety of different systems architectures [1] and they have been used extensively in the ATM literature to evaluate ATM architectures and concepts. Given the relevance of these methods to the UAM case study used in this dissertation, this section provides a brief overview of the different types of simulation studies in the ATM literature.

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 [65]. However, Odoni notes in [65] that these models do not adequately analyze safety.

The second category of models are 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 derived from control theory to model air traffic management as a control system. The goal of these models is 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 [66, 67, 68].

By contrast, human performance models were developed to model the cognitive functions and decision-making of air traffic controllers or flight crews. Because the current air traffic control system is so human-centric, human performance models were needed to analyze the impact of changes such as new ATM concepts or procedures on human performance metrics such as workload [69, 70, 71].

Finally, 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 [72, 73, 74, 75]. 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) [76, 77] 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, they are used to predict overall system behavior and performance.

Limitations of Quantitative Methods

While quantitative methods have been applied successfully in the aerospace and automotive industries to create and analyze physical or more detailed architectures, these methods are more challenging to apply when developing other types of architectures such as conceptual or functional architectures.

One key reason is that it is much harder to identify relevant quantitative metrics for evaluating architectures at the conceptual level compared to the physical level. For example, mass and cost are common quantitative metrics for evaluating a physical architecture. However, it is more challenging to identify suitable metrics to quantify the safety or security of a conceptual or functional architecture because safety and security are not physical properties.

Even if quantitative metrics exist, the performance of a conceptual or functional architecture with respect to a given metric is challenging to evaluate. This is because in the early stages of system design, many of the design details needed to calculate quantitative metrics are not yet known. For example, mass and cost can be calculated for a physical architecture but are much harder to calculate for a conceptual or functional that only defines a set of functions and the interactions between them, not the details of how they are implemented.

These issues suggest that creating system architectures, especially during the early stages of system design, is a fundamentally different type of problem solving activity than creating physical architectures and different techniques are required [78]. DeRemer and Kron make this observation in [79] when they state that:

Structuring a large collection of modules to form a "system" is an essentially distinct and different intellectual activity from that of constructing the individual modules [79, p. 80]

In summary, quantitative methods enable more systematic and thorough exploration of a tradespace compared to decomposition-based or reuse-based methods when creating physical system architectures. However, because they require architectures to be evaluated strictly in terms of quantitative metrics, it is challenging to use quantitative methods to evaluate conceptual or functional architectures for emergent properties like safety.

2.1.4 Flow-Based Methods

The last type of approach to architecture development is flow-based methods. Flow-based architecture development methods are typically used to design systems where the primary goal of the system involves flow-based properties such as efficiency or throughput. For example, a logistics network might be designed using flow-based methods to maximize the speed and efficiency with which packages can be transported from source to destination. Similarly, a communications network might be designed using flow-based methods to maximize the data throughput or the number of clients that can be served.

Flow-based methods model the system as a network of nodes (i.e., a graph) through which items, energy or data flow through. These graph-based models are then used to identify how best to link the nodes to achieve the desired properties. Flow-based methods have been used to design mission architectures and space logistics networks [80], utility, and transit infrastructure networks in a city [81] as well as software systems [31, 82].

Limitations of Flow-Based Methods

While these flow-based methods have been successfully used to design systems to achieve flow-based properties such as efficiency or throughput, they are not suitable for designing systems to achieve non-flow-based properties such as safety because they only focus on the flow interactions between nodes of the system. However, as discussed in Chapter 1, it is the control-oriented interactions in a system that give rise to emergent properties. Thus, like the object-oriented models used in decomposition-based methods, flow-based models do not focus enough on the essential control-oriented interactions that enable emergent properties such as safety to be achieved.

2.2 Limitations of Current Approaches

In section 2.1, a wide variety of architecture development methods were reviewed. Based on the limitations discussed for each category of methods, the key limitations that are addressed by this research are as follows.

First, the modeling approach underlying many of the architecture development methods do not model the control-oriented interactions in the system that are critical to ensuring that emergent properties such as safety and security are achieved. Object-oriented system models focus too much on the physical components and interfaces in a system while flow-based system models focus too much on flow interactions. As a result, it is much harder for system designers to reason about the necessary functions and control interactions that should be included in the system design because they are obscured by these system models.

Second, although many of the architecture development methods recognize the importance of deciding how to divide up a system into components, they typically offer little guidance on how to make that decision for a specific system. Some methods (e.g., MBSE methods) offer a variety of heuristics that a systems engineer can use to inform how they decompose their system. Other methods rely on the use of patterns or reference architectures that a system designer can customize like a template. In either case, however, little guidance is provided on how to decide which heuristics to apply and there is heavy reliance on the experience of the system designer to make these decisions. Furthermore, for new or novel systems that have never been designed before, useful heuristics, patterns or reference architectures may not exist yet because a design precedent has not yet been established.

Third, many of the current architecture development methods evaluate architecture options using methods that are not suitable for conceptual or logical architectures created at the early stages of system design. This limitation arises because these methods require architecture options to be evaluated using quantitative criteria that are easiest to identify when creating a physical system architecture. However, some desirable emergent properties do not have associated quantitative criteria that can be easily defined. Furthermore, even if quantitative criteria are identified, it is much harder to quantitatively evaluate the performance of early-stage conceptual or logical architectures with respect to those criteria because many of the design details have not yet been decided.

These limitations thus suggest that there is a need to develop a more structured and systematic approach for designing emergent properties into system architectures that can be applied starting at the early stages of system design. This research proposes using Systems Theory as the foundation for this new approach.

2.3 Introduction to Systems Theory

To design emergent properties into modern complex systems, the design process needs to avoid the limitations discussed in Sections 2.1 and 2.2 by using an approach that is focused on the control-oriented aspects of a system and emphasizes the need to consider the system as a whole. Instead of relying on decomposition, architecture development needs to take a holistic control-oriented approach based on systems theory.

As discussed in Chapter 1, architecting modern complex systems to achieve emergent properties such as safety is challenging because they are becoming more complex and interconnected. Because of the increased coupling between components, their behavior is no longer independent and depends on both the inputs received from other components as well as the context or environment they are operating in. Unfortunately, decomposition-based approaches overemphasize the independence of the components [83, 84], making it more difficult to identify or analyze the emergent behaviors or properties of the system [2, 8, 84].

Systems theory was created in response to this need to view systems more holistically. It recognizes that the properties or behaviors of the system are not just the sum of the behaviors of the components and that the system's behavior depends on the environment in which it operates [83, 85]. For this reason, Systems Theory emphasizes considering the system as a whole, including the context or environment in which the system operates.

Systems theory also recognizes that the behavior of a system arises from circular loops of cause-and-effect relationships instead of linear chains. As Peter Senge states, "reality is made up of circles but we see straight lines" [86, p. 73]. Instead of understanding the behavior of a system in terms of one event leading to another (a linear view of causality), Systems Theory views a system's behavior as being influenced by feedback in continuously operating circular loops [86, 87]. As a result of these circular loops of cause and effect, behavior in one part of the system can eventually influence another part of the system even if they are not directly coupled or connected to each other [45]. For this reason, the behavior of a system arises from the structure of its control and feedback loops [88].

There are two pairs of key concepts that form the foundation of Systems Theory: Hierarchy and Emergence, and Communication and Control [2, 8]. First, in systems theory, a system can be organized into hierarchical levels such that the properties associated with the system elements at one level arise (i.e., emerge) from the interactions between the parts at the next lower level [2, 8]. Extending these ideas to engineered systems, any complex system can also be organized into hierarchies of subsystems, functions or components. Emergent properties such as safety thus arise from the interactions between the system components at the level below [2].

This leads to the second pair of key concepts: Communication and Control. To ensure that the necessary interactions between system components occur, components at one level of the hierarchy can apply controls onto the level below to constrain the interactions that occur at the lower level [2, 8]. The enforcement of these constraints thus ensures that the required interactions occur and undesirable interactions are avoided. In addition, the implementation of these controls requires the communication of both controls down to the components below as well as feedback up to the controller. Communication and control are therefore the means by which constraints on a system's behavior are enforced [2].

2.4 Overview of STAMP and STPA

Systems-Theoretic Accident Model and Processes (STAMP) is an accident causality model that is based on Systems Theory [2]. As in Systems Theory, STAMP emphasizes the importance of considering the system as a whole. This includes not just the technical aspects such as hardware and software but also the human operators, the social and organizational aspects as well as the system's operating environment. In addition, STAMP also recognizes that emergent properties such as safety arise from the interactions between the system components. This holistic view of a system enables STAMP to explain how accidents or undesirable behavior might occur due to non-linear or indirect causes, design and requirements flaws, and human factors issues in addition to component failure.

STAMP also treats safety as a control problem rather than a reliability problem. Instead of focusing on preventing component failure, STAMP focuses on preventing accidents or unacceptable losses by ensuring the necessary interactions and behaviors occur and undesirable interactions or behaviors are avoided. This can be done by identifying and enforcing sufficient constraints on the system's behavior and the interactions between the system components.

Based on this concept of safety as a control problem, STAMP models the controls in a system using a hierarchical safety control structure that contains a controlled process and the various controllers that can influence or control the system's behavior. This is illustrated in Figure 4.

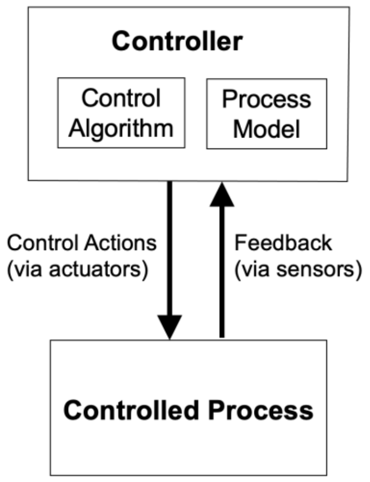


Figure 4: A simple control loop (from [89])

Under this paradigm, a controller enforces the system constraints by applying appropriate control actions to control a system's behavior or the interactions between its components. In turn, the controller receives feedback about the effect of those controls on the system. This concept of control is interpreted broadly. Although the controls could be technical or physical controls, they may also be social or organizational controls.

Process models are another important and unique aspect of STAMP. Process models (also known as mental models for humans) are important for the safe operation of a system because they are used by controllers to make decisions and select appropriate control actions. For this reason, controllers must receive adequate feedback to keep process models updated over time to avoid making unsafe decisions based on an incorrect process model. For example, if a pilot's mental model of their aircraft is inconsistent with the actual aircraft state, they may provide control inputs that are unsafe in the context of the actual state of the aircraft.

Based on this theoretical foundation, a hazard analysis technique called Systems-Theoretic Process Analysis (STPA) was created. STPA takes a more generalized view of accidents and losses. Although a loss may involve human death or injury, it may also involve other types of losses such as equipment, mission, financial or information losses. This enables a wide variety of control-oriented emergent properties to be analyzed using STPA including maintainability [90] and scalability [91]. Figure 5 shows the four steps in STPA.

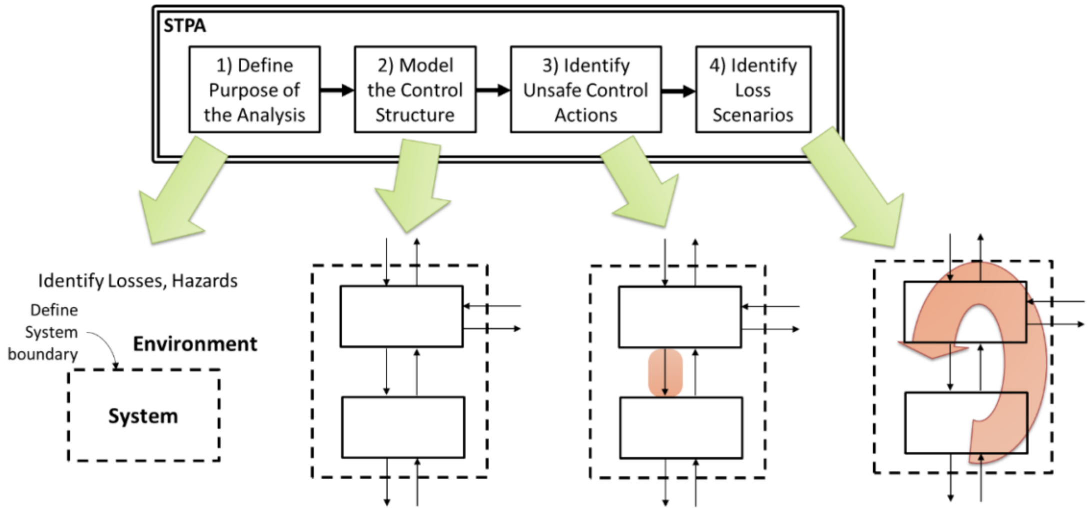

STPA
1) Define Purpose of the Analysis
2) Model the Control Structure
3) Identify Unsafe Control Actions
4) Identify Loss Scenarios
Identify Losses, Hazards
Define System boundary
Environment
System

Figure 5: The STPA process (from [89])

STPA analyzes the control loops in a safety control structure to proactively identify potential flaws and causes of accidents during development before an actual accident occurs [89]. These flaws and causal factors are identified as Unsafe Control Actions (UCAs) and causal scenarios.

Because of STPA's focus on identifying potential flaws in control loops, an STPA analysis can be used to inform how a system should be designed or how to improve an existing design to mitigate or prevent the UCAs and scenarios. However, a more structured process is needed for using the STPA results to create and assess architecture options.

2.5 Past Research Using STAMP and STPA for Architecture Development

In addition to the safety-driven approach to architecture development that was developed in [15], there have been several other research efforts that have also applied STAMP and STPA to architecture development. This section provides a brief overview of these past research efforts to highlight some specific aspects of architecture creation and assessment that a systems-theoretic architecture development framework should address.

Comparing Architecture Options Using STPA

One way that STPA has been used in architecture development is to analyze a series of architecture options that have already been created and compare the results. For example, Kharsansky used STPA to compare three architecture options for controlling and managing a constellation of satellites in terms of the reliability and safety of the architecture as well as the ability to scale the architecture to larger constellation sizes [91].

Similarly, France used STPA to compare four architecture options for an automated park assist system where each architecture option gives the automation an increasing level of control over the vehicle and the parking task [92]. France then compared the different architectures in terms of the number of driver and automation UCAs identified for each option [92].

As a final example, Horney used STPA to analyze two architecture options for controlling the formation shape of one or more unmanned aircraft that are tethered to a lead human-piloted aircraft [93]. In one option, the human pilot decided the formation shape and in the other, the tethered unmanned aircraft decided the formation shape. Horney then compared the two options in terms of the identified UCAs and scenarios and used them to highlight the potential challenges of each option [93].

These past research efforts all employ a common strategy for comparing architecture options. They identify a series of architecture options first, evaluate each one with respect to a set of pre-determined criteria (e.g., number of UCAs or scenarios) and then compare them based on those criteria to determine the benefits and tradeoffs of different options.

Although these research efforts are an improvement over the traditional methods for architecture development, there are two key limitations. First, more guidance is needed on how to systematically identify the architecture options to be evaluated instead of just using heuristics or experience. For example, Kharsansky and France defined architecture options in terms of different levels of automation, a heuristic that is commonly used when deciding how much automation to include in a system. Instead of just relying on heuristics or past experience, a more systematic process is needed for identifying what architecture options should be considered.

Second, more guidance is needed on identifying the criteria by which architecture options should be assessed. Instead of just comparing architecture options in terms of the number of different types of UCAs or scenarios identified, a more structured process is needed to help identify appropriate metrics of interest for a specific system.

Using STPA to Improve an Architecture

Another way that STPA has been used in architecture development is to analyze an initial architecture using STPA and use the results to inform changes to improve that architecture. When done iteratively, the architecture can be improved incrementally.

One method that does this is called Systems-Theoretic Early Concept Analysis (STECA) [94]. STECA is based on systems theory and extends STAMP and STPA to analyze the Concept of Operations (ConOps) for a system early in the design process. STECA focuses on systematically identifying missing information, undocumented assumptions and inconsistent or conflicting information in the ConOps and formulating mitigation strategies to address these problems [94].

To do this, STECA first models the system based on the ConOps using a control structure. It then defines a set of formal equations that can be used to analyze the control structure for gaps in a mathematically rigorous manner. Three main gaps are analyzed: (1) completeness in the definition of the control loops, (2) constraints fully accounted for, and (3) consistency and clarity where responsibilities or control actions are shared by multiple controllers in the control structure [94]. STECA then provides a process for modifying the control structure to remedy any gaps that are identified. The STECA analysis and design process is illustrated in Figure 6.

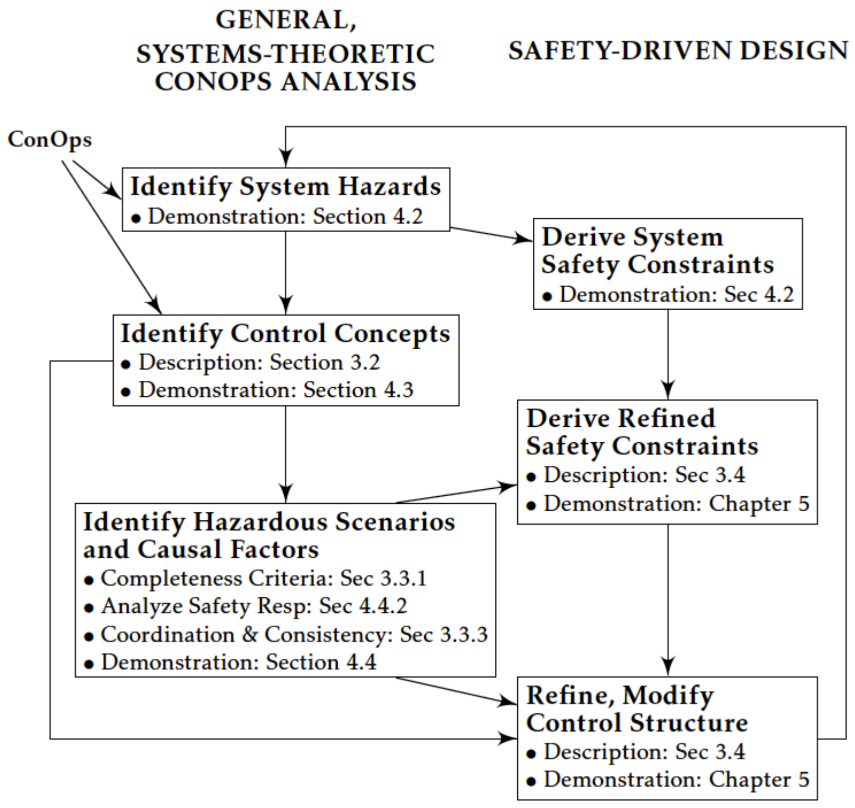

GENERAL, SYSTEMS-THEORETIC CONOPS ANALYSIS
SAFETY-DRIVEN DESIGN
ConOps
Identify System Hazards
• Demonstration: Section 4.2
Derive System Safety Constraints
• Demonstration: Sec 4.2
Identify Control Concepts
• Description: Section 3.2
• Demonstration: Section 4.3
Derive Refined Safety Constraints
• Description: Sec 3.4
• Demonstration: Chapter 5
Identify Hazardous Scenarios and Causal Factors
• Completeness Criteria: Sec 3.3.1
• Analyze Safety Resp: Sec 4.4.2
• Coordination & Consistency: Sec 3.3.3
• Demonstration: Section 4.4
Refine, Modify Control Structure
• Description: Sec 3.4
• Demonstration: Chapter 5

Figure 6: STECA process flow diagram (from [94])

Another method that uses the results of an STPA analysis to improve an initial system architecture is the conceptual architecture-based approach described by Leveson in [14]. In this design process, an initial system architecture (called a conceptual architecture) is analyzed using STPA to identify UCAs and scenarios that describe potential causes of unsafe system behavior. Changes to the system architecture can then be identified that will mitigate or eliminate the identified UCAs or scenarios and the STPA analysis can be updated to reflect the new version of the system architecture. In [14], this process is applied to the design of a Thermal Tile Processing System (TTPS) robot.

The TTPS robot is an automated vehicle that was intended to be used to refurbish thermal tiles on the space shuttle after a space flight. In essence, the robot consisted of a mobile base to move from one location to another and a robotic arm that serviced the thermal tiles on the space shuttle. To prevent the mobile base from tipping over while the robotic arm was extended, the mobile base included stabilizer legs that needed to be deployed and secured before extending the robotic arm [14]. An initial architecture for this robot used separate controllers to control the movement of the robotic arm and stabilizer legs as shown in Figure 7.

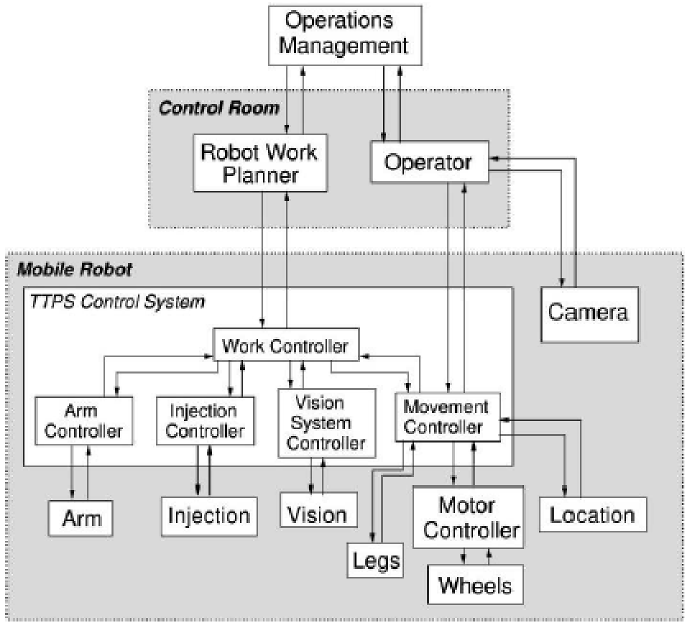


Figure 7: Initial system architecture for TTPS robot (from [14])

An STPA analysis of this initial architecture found that poor coordination between the two controllers controlling the robotic arm and stabilizer legs was associated with numerous hazardous scenarios. For example, the stabilizer legs could be retracted before the robotic arm was fully stowed or the robotic arm could be extended before the stabilizer legs were fully deployed, either of which could cause the robot to tip over.

Based on these STPA results, Leveson finds that many of these hazardous scenarios could be prevented by using the same controller to control the stabilizer legs and robotic arm instead of using separate controllers [14]. Having the same controller be responsible for controlling both the stabilizer legs and robotic arm makes it easier to coordinate their movements. This alternative architecture is shown in Figure 8.

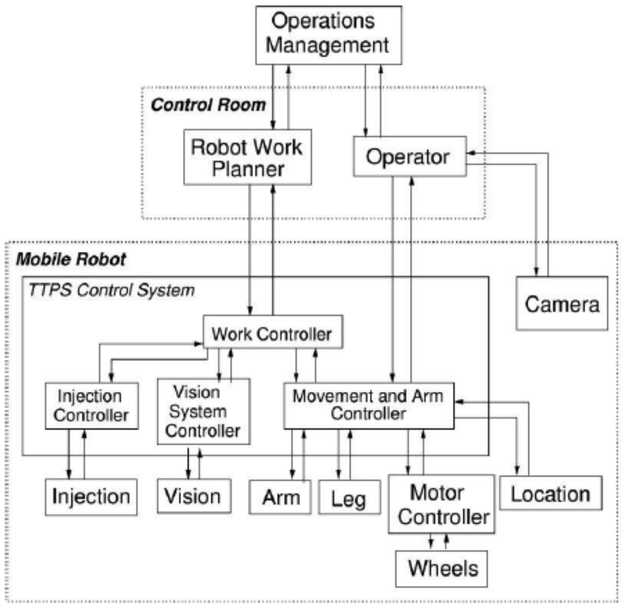


Figure 8: Alternative architecture for the TTPS robot (from [14])

This example illustrates how information about how a system might behave can inform the development of its architecture. Because the STPA analysis identified that coordinating the movement of the robotic arm and stabilizer legs would be critical for ensuring safe system operation, a better architecture was identified that used the same controller to control both parts. This functional grouping makes it easier to ensure that unsafe behavior would be avoided.

Both STECA and the conceptual architecture-based approach are useful because they help systems engineers to identify information that can inform changes to the architecture. By helping to highlight flaws or inconsistencies in a system architecture, both STECA and the conceptual architecture-based approach provide systems engineers with useful information that can guide and inform their design decisions. It is this type of design support that a safety-driven architecture development framework should also strive to provide.

However, both STECA and the conceptual architecture-based approach require an initial ConOps or conceptual architecture to be defined first to perform the initial analysis on. That initial architecture is then modified to address any flaws that are found. Instead of creating an initial architecture and then addressing any flaws, it would be preferable to create an initial architecture that avoids as many of the flaws as possible from the beginning.

2.6 Summary

This chapter surveyed a wide variety of architecture development methods and identified several key limitations to be addressed. First, many of these methods do not focus enough on the control-oriented interactions that are critical to ensuring that emergent properties such as safety are achieved. Second, they rely primarily on general heuristics to guide the creation of a system architecture and typically offer little guidance on how to make those design decisions for a specific system. Finally, they rely on quantitative criteria for comparing architecture options even though it can be difficult to identify appropriate quantitative criteria for emergent properties like safety, especially during the early stages of development.

Instead of these traditional approaches, systems theory and STAMP offer a more suitable approach for designing emergent properties like safety into a system architecture from the beginning. Because of STPA's focus on identifying potential flaws in control loops, an STPA analysis provides useful information that can be used to inform decisions about how a system should be architected to achieve safety and other emergent properties. The next chapter describes the architecture development framework that was developed to structure the process of using STPA results to inform architectural design decisions and create a system architecture that best achieves the desired emergent properties.

Chapter 3 A Safety-Driven Approach to Architecture Development

As described in Chapter 1, this research aims to develop a framework for architecture development that enables systems engineers to design emergent properties like safety into their system architectures. To address the limitations of current methods that were described in Chapter 2, this framework will need to provide three main types of support. First, instead of simply relying on decomposition to identify the system elements, this framework needs to help systems engineers reason about what functions and interactions will need to be included in the system architecture to achieve the desired emergent properties. Second, instead of just relying on design heuristics, this framework needs to help generate relevant design information that systems engineers can use to inform their architectural design decisions. Finally, instead of only using quantitative metrics to evaluate and compare architecture options, this framework needs to help identify relevant evaluation criteria that systems engineers and analysts can use to determine how well an architecture option achieves the desired emergent properties. Because this framework is focused on early-stage architecture development, it is also important that these criteria can be identified even when few details about the system are known.

To meet these needs and enable systems engineers to design emergent properties into systems, Systems Theory provides a suitable theoretical foundation for this framework. This chapter is organized as follows. First, the concepts from Systems Theory (described in Section 2.3) are applied to define a systems-theoretic approach to architecture development. Then, an overview of the framework is provided followed by the details of how each part of the framework was developed.

3.1 A Systems-Theoretic Approach to Architecture Development

In any architecture development effort, the overarching goal is to determine how the system should be designed to achieve the desired emergent properties while avoiding undesirable behavior. Ultimately, this requires deciding what functions the system needs to perform, what interactions are needed between functions, what the components of the system should be and how they should be structured. To create a systems-theoretic approach to architecture development, the concepts from systems theory can be applied to each of these design decisions.

As discussed in Section 2.3, one of the key concepts from Systems Theory is Holism – the idea that the behavior of a system depends on the context it operates in. Applying this concept to architecture development therefore suggests that systems need to be designed as a whole. This means that design decisions should account for both the interactions between functions or components of the system as well as the interactions between the system and the environment in which it operates. These interactions are an especially important aspect of the system design because the desired emergent properties of a system can only be achieved if the necessary interactions are designed into a system while avoiding undesirable ones.

Similarly, applying the concepts of hierarchy and emergence as well as communication and control suggests emergent properties can be designed into a system by enforcing sufficient constraints to control the behavior of the system components and the interactions between them. This requires that the system design contains the right components arranged in an adequate hierarchy with the necessary communication (feedback and control actions) to achieve the system goals while enforcing constraints on how those goals can be achieved.

A system-level design process should therefore assist system designers in creating the system architecture by helping them to identify the necessary interactions (e.g., control actions and feedback) and the required system structure. This will ensure that components at a particular level of the control hierarchy are designed to exert adequate control over the components at the level below.

3.2 Overview of the Safety-Driven Architecture Development Framework

Based on the overall approach described in the previous section, a new framework for developing system architectures called the *Safety-Driven Architecture Development Framework (SDADF)* was developed to help systems engineers design emergent properties like safety into a system architecture from the beginning of architecture development. Conceptually, the framework consists of 3 main parts as illustrated in Figure 9.

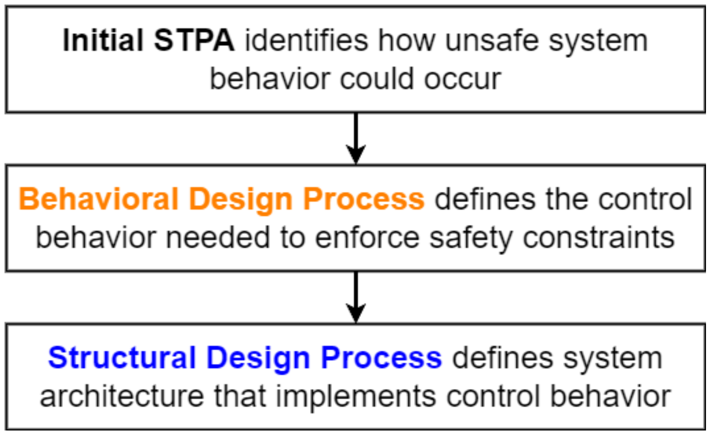

Initial STPA identifies how unsafe system behavior could occur
Behavioral Design Process defines the control behavior needed to enforce safety constraints
Structural Design Process defines system architecture that implements control behavior

Figure 9: Conceptual overview of safety-driven architecture development framework

The key overarching idea is that a system should be designed to prevent unsafe or undesirable behavior. Thus, the first part of this framework is to perform an initial STPA analysis of the system to identify preliminary information about how unsafe behavior of the system might occur. One of the strengths of STPA is that it analyzes a system including the context in which that system operates. Thus, using STPA results to drive design decisions ensures that those design decisions account for the system's operating context.

This framework applies the existing STPA process with no changes. However, because few design details are known during early-stage design, this initial STPA is performed at a high-level of abstraction to minimize the number of assumptions that need to be made about the system during the analysis. This initial abstract definition of the system can then be refined as architecture development progresses.

Once potential unsafe system behaviors have been identified, the next part of the framework defines the control behavior needed to prevent those unsafe behaviors. Developing the control behavior before exploring and comparing architecture options allows systems engineers to determine what the desired behavior of the system should be before creating a system structure to implement it. Thus, the behavioral design of a system serves as a cognitive steppingstone to support later reasoning about what the preferred system architecture might be.

A *Behavioral Design Process* was therefore developed to provide a structured process for using the causal scenarios identified by STPA to define the necessary safety constraints and the control loops that are needed to enforce them. The output of this part of the framework is a *Conceptual Architecture*, a control-oriented system model that represents the various control elements that are needed in the system and the relationships between them.

Once the desired control behavior has been defined, the final part of this framework involves creating a system architecture to implement that desired behavior. To do this, a *Structural Design Process* was developed to provide a systematic process for deciding how to allocate the control elements to either new or existing system components to create the system architecture. Because there can be numerous options for how to allocate the control elements to achieve the same desired behavior, this process helps systems engineers to systematically explore and compare different architecture options to identify the one that best achieves the desired emergent properties.

The remainder of this chapter elaborates on the details of the behavioral and structural design processes. This framework is then applied to develop and refine an ATM architecture for UAM in Chapter 4 and Chapter 5.

3.3 The Behavioral Design Process

The purpose of the behavioral design process is to define the control behavior that is needed to enforce the necessary safety constraints and ensure unsafe or undesirable behavior is prevented. However, designing an adequate control behavior can be difficult to do because modern complex systems typically require numerous interdependent safety constraints to be enforced. The behavioral design may therefore need to contain many control functions and interactions to adequately enforce all the safety constraints. Furthermore, the interdependence between control functions makes it difficult to ensure that design decisions made to avoid one type of unsafe behavior do not inadvertently lead to another.

For these reasons, a structured and iterative process is needed to help systems engineers incrementally refine the required control behavior and evaluate it to ensure that flaws are not introduced as the behavior is designed. Figure 10 provides an overview of the behavioral design process.

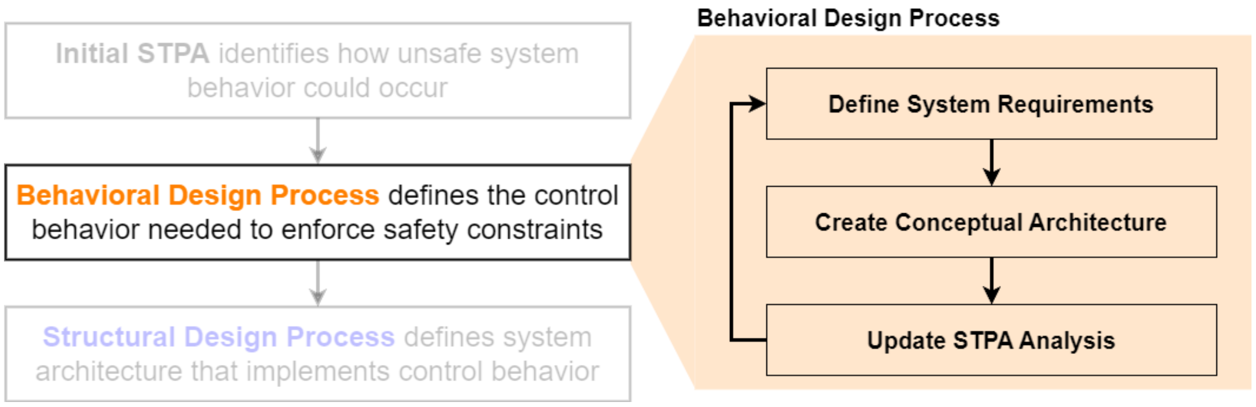

Initial STPA identifies how unsafe system behavior could occur
Behavioral Design Process defines the control behavior needed to enforce safety constraints
Structural Design Process defines system architecture that implements control behavior
Behavioral Design Process
Define System Requirements
Create Conceptual Architecture
Update STPA Analysis

Figure 10: Overview of behavioral design process to define required control loops

3.3.1 Defining System Requirements

The input to this behavioral design process is the causal scenarios identified by the initial STPA. As shown in Figure 10, this process starts by using those scenarios to identify appropriate system requirements. Consistent with STAMP principles, the system requirements define the safety constraints that will need to be enforced to prevent the unsafe behaviors described in the STPA causal scenarios. These requirements are intended to be solution-neutral and should only state what constraint(s) need to be enforced. The requirements should not describe how the constraints should be implemented or who should enforce the constraints because those decisions will be made later in the development process when additional design information is available to make a more informed decision.

As an example, consider a simple, abstracted version of an Air Traffic Control (ATC) system where *Air Traffic Management* is a controller that monitors the movement of aircraft in the airspace and issues a *Coordination* control action to prevent collisions by coordinating the movement of aircraft. The control structure for this simple ATC system is shown in Figure 11.

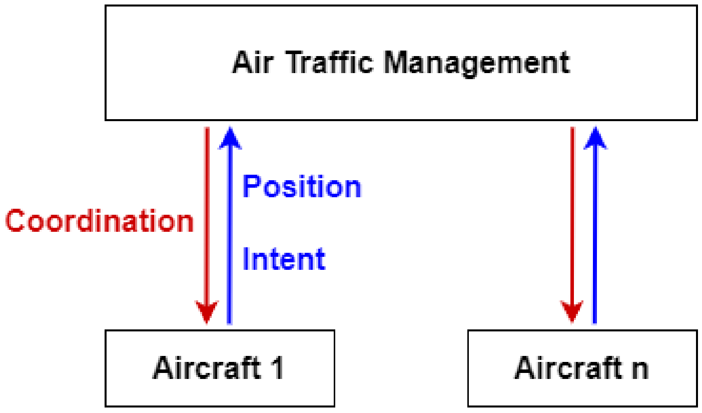
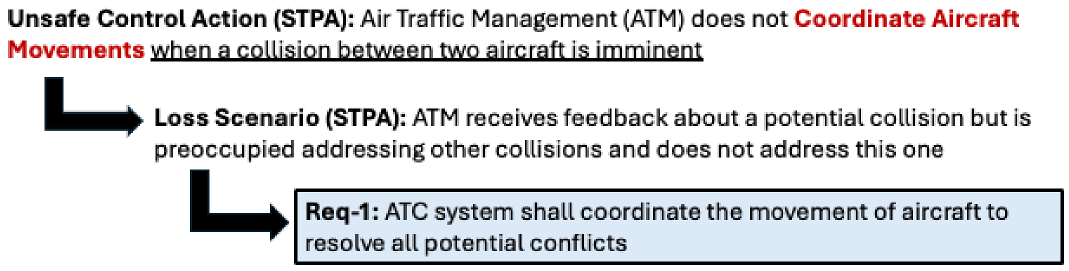


Figure 11: A simple control structure of the air traffic control system

Figure 12 shows an example of how the *Coordination* control action can be analyzed to derive a collision avoidance requirement, Req-1.



Unsafe Control Action (STPA): Air Traffic Management (ATM) does not **Coordinate Aircraft Movements** when a collision between two aircraft is imminent

Loss Scenario (STPA): ATM receives feedback about a potential collision but is preoccupied addressing other collisions and does not address this one

Req-1: ATC system shall coordinate the movement of aircraft to resolve all potential conflicts

Figure 12: Example requirement derived from initial STPA analysis (control action in red)

As illustrated in Figure 12, each requirement defines a constraint that, when enforced in the system, would prevent or mitigate one or more scenarios. By doing this for all the scenarios identified in the initial STPA analysis, a set of system requirements are defined that, when implemented, will adequately control the hazards.

3.3.2 Creating the Conceptual Architecture

Once the system requirements have been defined, a *Conceptual Architecture* is created to model the control loops that are needed to enforce the safety constraints described in the requirements. Unlike the system architecture which models the physical components and relationships between them, the conceptual architecture does not necessarily model the physical components. Instead, it is a functional control structure that models the control behavior that the system will need to exhibit in terms of the required control elements and the relationships between them.

Inspired by the elements of a basic control loop, a conceptual architecture includes four main types of control elements as shown in Figure 13. The creation of each control element is therefore a design decision that needs to be made. By defining these four types of control elements, adequate control loops can be created to enforce the safety constraints described in the requirements.

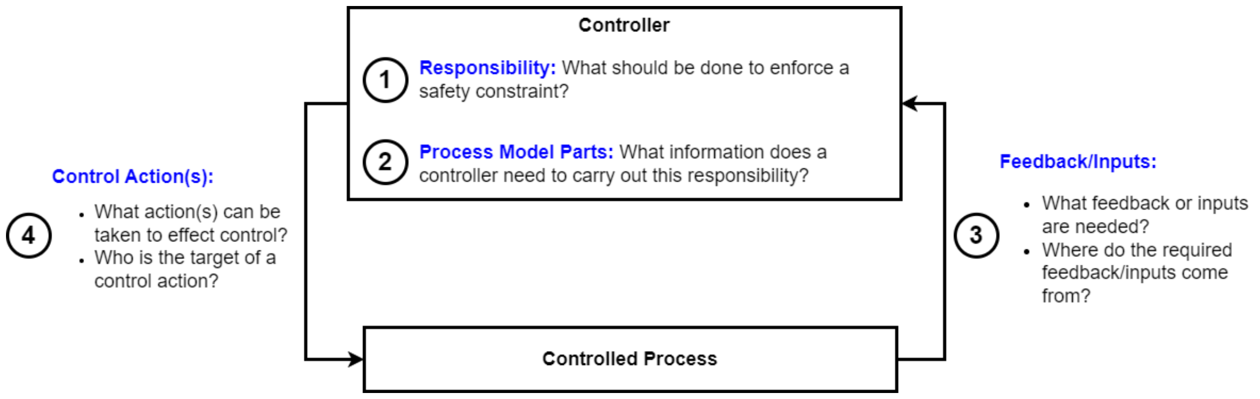

Controller
1
Responsibility: What should be done to enforce a safety constraint?
2
Process Model Parts: What information does a controller need to carry out this responsibility?
Control Action(s):
4
What action(s) can be taken to effect control?
Who is the target of a control action?
Feedback/Inputs:
3
What feedback or inputs are needed?
Where do the required feedback/inputs come from?
Controlled Process

Figure 13: Illustration of the four types of control elements in a conceptual architecture

Defining Control Responsibilities (Control Element 1)

Creating a conceptual architecture starts with identifying the control responsibilities that that system will need to perform. To ensure that all safety constraints described in the requirements are enforced, these control responsibilities are derived from the system requirements. However, not every system requirement generates a new control responsibility because a single control responsibility may have multiple system requirements that specify different aspects of its required behavior. For example, one responsibility of the ATM system is to prevent collisions between aircraft. However, to specify how this responsibility should be carried out, numerous requirements are needed to describe what feedback is needed, the factors that should be considered when modifying the path of an aircraft to prevent a collision, and how quickly those decisions should be made.

For this reason, groups of related system requirements are used to derive the control responsibilities and their corresponding constraints that specify restrictions on how those responsibilities should be performed. Figure 14 illustrates how this is done for eight generic system requirements to generate two control responsibilities and six constraints.

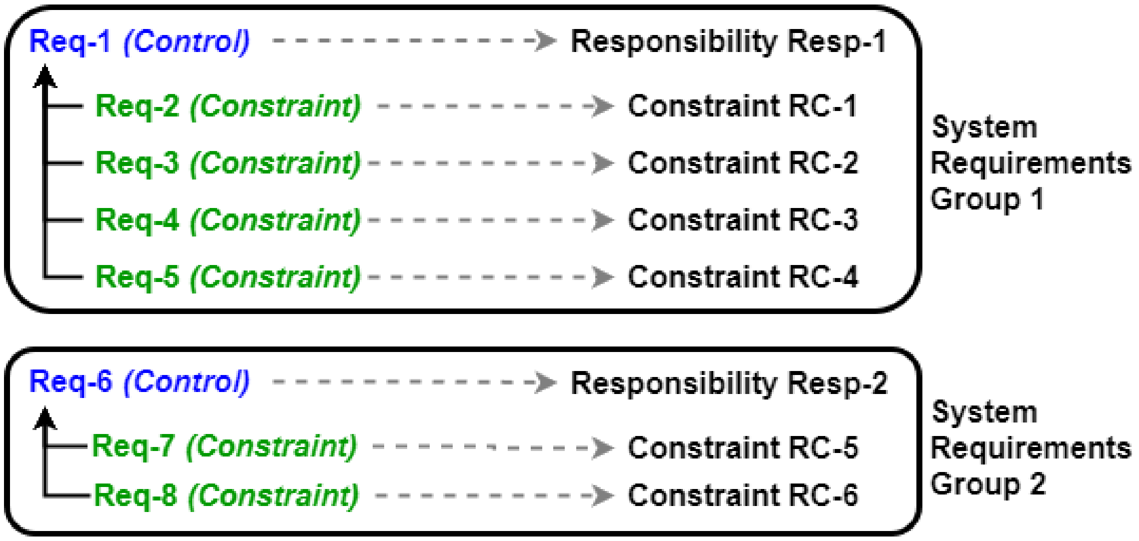


Figure 14: Deriving control responsibilities and constraints from system requirements

To form these requirement groups, each system requirement is first categorized as either a control requirement or a constraint requirement. Control requirements describe a control decision or control function that needs to be performed. By contrast, constraint requirements describe restrictions or constraints on acceptable ways that a control decision should be made or the expected response of the controlled process in the system.

Once the system requirements are classified, they can then be organized into groups where each group consists of one control requirement and the constraint requirements that apply to it. This is illustrated by the blue and green requirements on the left side of Figure 14. Grouping the requirements like this ensures that related requirements are considered together when the control behavior is developed. For each group of requirements, the control requirement is used to generate a control responsibility, and the constraint requirements are used to generate responsibility constraints (RCs) that are associated with the control responsibility.

Continuing the simple ATC system example illustrated in Figure 11, consider the three system requirements shown in Table 1 that describe several aspects of how air traffic should be managed to prevent collisions. To the right of each requirement is its classification.

Table 1: Example classification of system requirements

Requirement	**Category**
Req-1: ATC system shall coordinate the movement of aircraft to resolve potential conflicts	**Control Requirement**
Req-2: ATC system shall account for operational constraints when selecting coordination	**Constraint Requirements**
Req-3: ATC system shall ensure that aircraft have received the coordination being communicated	

All three of the requirements in Table 1 pertain to the same aspect of air traffic control: coordinating the movement of aircraft to prevent collisions. However, Req-1 describes the control decision to be made (resolving potential conflicts) while Req-2 and Req-3 describe constraints on the inputs that should be considered when making that decision.

Using the three requirements shown in Table 1, a control responsibility (Resp-1) for preventing conflicts and its associated behavioral constraints (RC-1 and RC-2) can be generated as shown in Table 2. The system requirement used to generate each responsibility or constraint is linked in red.

Table 2: Example control responsibility and constraints

Control Responsibility	**Resp-1:** Coordinate the movement of aircraft to prevent conflicts *[Req-1]*
Constraints	**RC-1:** Account for operational constraints when selecting coordination *[Req-2]* **RC-2:** Ensure that aircraft have received the coordination being communicated *[Req-3]*

Defining Process Model Parts, Control Actions and Feedback (Control Elements 2, 3 and 4)

Once the control responsibilities and associated constraints have been generated from the system requirements, the process model parts, control actions, and feedback can then be defined based on what is needed to carry out each responsibility and meet its behavioral constraints. Figure 15 illustrates how this is done.

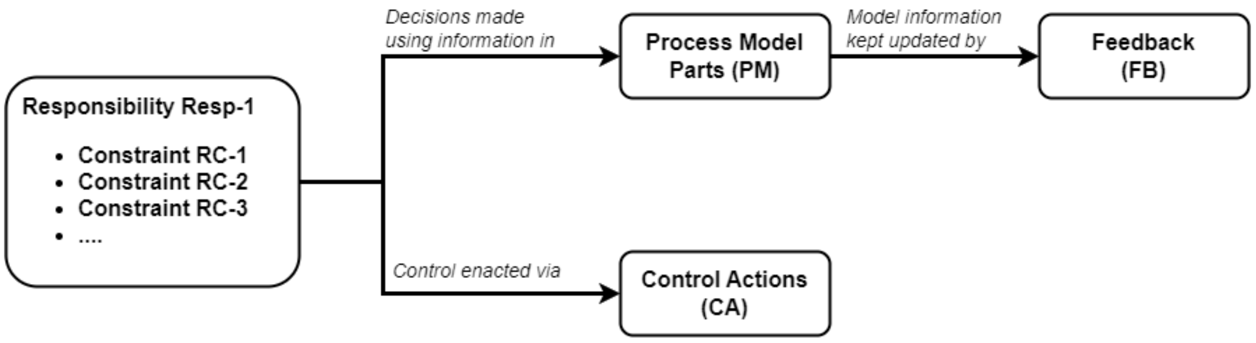


Figure 15: Identifying the other control elements from responsibilities and constraints

As illustrated in Figure 15, the responsibilities and associated constraints are first used to generate the process model parts and control actions. Process model parts contain the information needed by a controller to make appropriate decisions when carrying out a responsibility. Thus, the process model parts for a given responsibility can be generated by considering what information will be needed to make the decision described by that responsibility and its associated constraints. Similarly, the control actions can be generated by considering what output(s) might be needed by that responsibility to enable effective control.

As described in STAMP, in addition to having the right information in the process model to make appropriate decisions, it is also important that those process model parts are kept updated over time and this requires appropriate feedback. Thus, for each process model part, the necessary feedback required to keep it updated should be identified.

As a concrete example of how process model parts, control actions, and feedback are defined, Figure 16 shows how these control elements are generated for Resp-1, RC-1, and RC-2 that were shown in Table 2. To continue maintaining traceability, the responsibility or constraint that was used to identify each control element is linked in red.

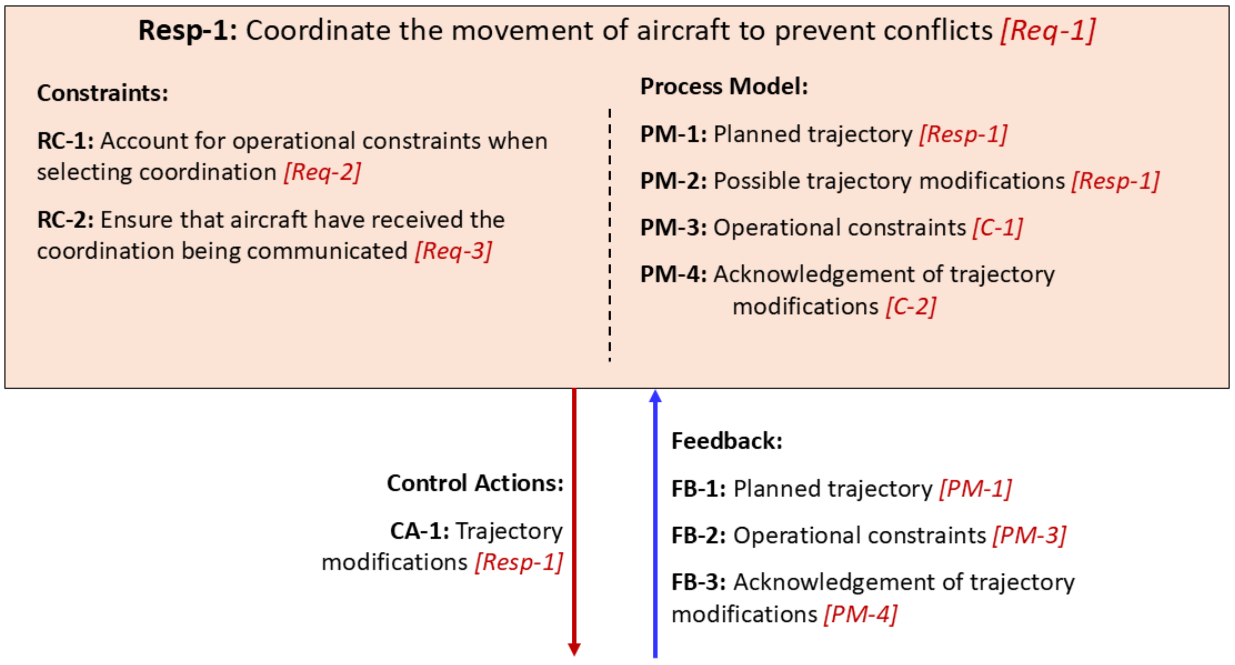

Resp-1: Coordinate the movement of aircraft to prevent conflicts [Req-1]
Constraints:
RC-1: Account for operational constraints when selecting coordination [Req-2]
RC-2: Ensure that aircraft have received the coordination being communicated [Req-3]
Process Model:
PM-1: Planned trajectory [Resp-1]
PM-2: Possible trajectory modifications [Resp-1]
PM-3: Operational constraints [C-1]
PM-4: Acknowledgement of trajectory modifications [C-2]
Control Actions:
CA-1: Trajectory modifications [Resp-1]
Feedback:
FB-1: Planned trajectory [PM-1]
FB-2: Operational constraints [PM-3]
FB-3: Acknowledgement of trajectory modifications [PM-4]

Figure 16: Example control elements generated for responsibility Resp-1

As shown in Figure 16, coordinating the movement of aircraft (Resp-1) requires identifying a conflict based on the planned trajectories of aircraft in the airspace (PM-1, updated by FB-1) and then modifying those trajectories (CA-1) to resolve that conflict. In addition, to account for operational constraints (RC-1), Resp-1 must know what they are (PM-3, updated by FB-2). Similarly, to ensure aircraft have received their trajectory modifications (RC-2), Resp-1 must receive confirmation that the aircraft received the trajectory modification (PM-4, updated by FB-3). This example therefore illustrates that this process allows systems engineers to carefully define a control loop for each responsibility by deciding the various control actions and feedback that are needed to carry out each responsibility and meet its associated constraints.

Defining Control Action Targets and Feedback Sources

By following this process for each of the defined responsibilities, the process model parts, control actions, and feedback associated with each responsibility can be generated. However, they cannot be assembled into a conceptual architecture yet until the targets of each control action and the sources of each piece of feedback are defined. To define the control action targets and feedback sources, the process model parts and constraints associated with the various responsibilities can be compared and the following rules can be applied:

- **Feedback sources** are the responsibilities or controlled process that have the required information in their process model
- **Control action targets** are the responsibilities or controlled process whose decision making must include the information in that control action

By applying these rules to the control actions and feedback associated with each responsibility, the relationships between responsibilities (and the controlled process) can be defined in terms of the control actions and feedback that are exchanged between them. Figure 17 illustrates how this is done for three generic responsibilities.

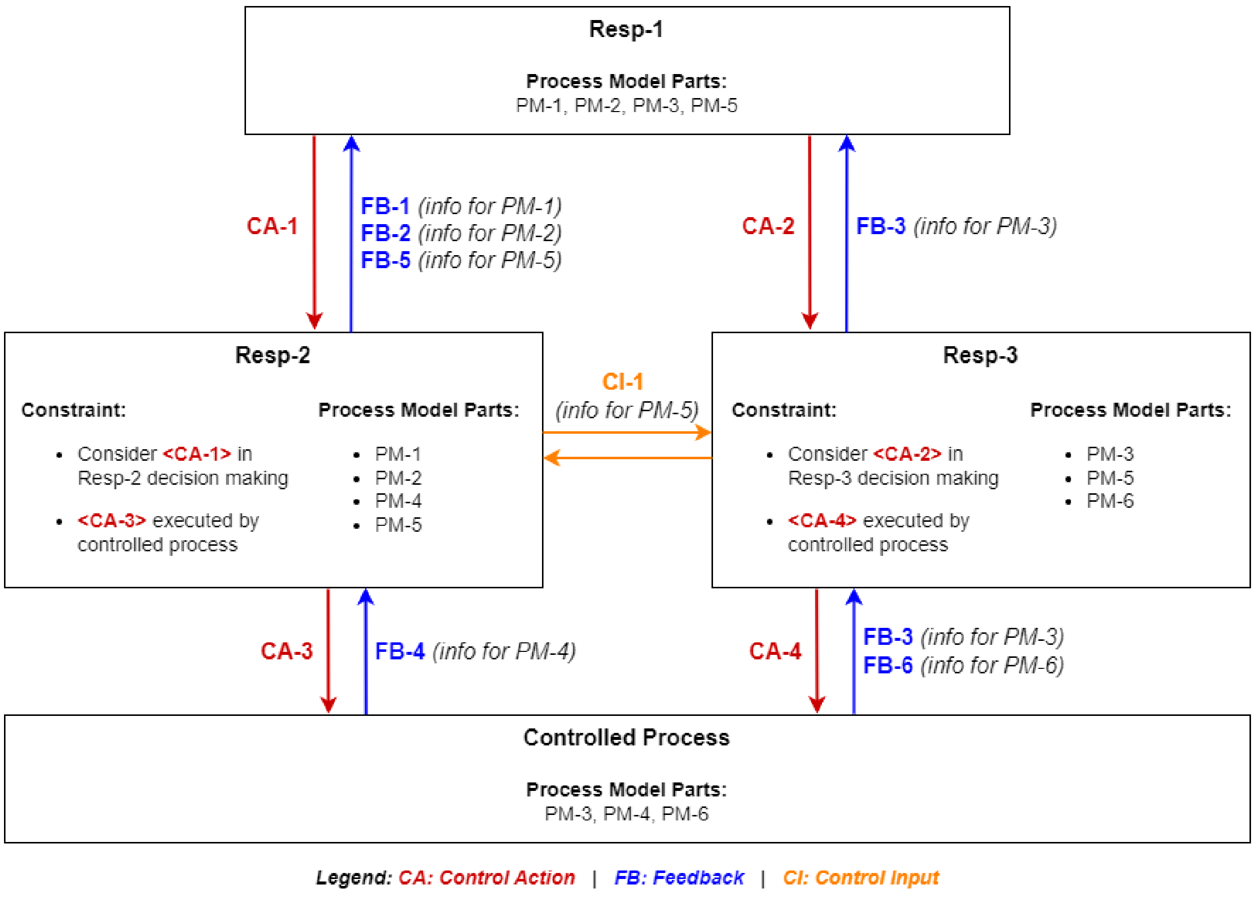


Figure 17: Defining control action targets and feedback sources

By applying the feedback sources rule, two responsibilities (or a responsibility and the controlled process) will be connected by feedback if they share the same process model part. Thus, in Figure 17, Resp-1 receives feedback FB-1, FB-2, and FB-5 from Resp-2 because those two responsibilities share process model parts PM-1, PM-2, and PM-5. Similarly, Resp-2 receives feedback FB-4 from the controlled process because the controlled process and Resp-2 share process model part PM-4. This same reasoning is also how the feedback sources were determined for the feedback between Resp-3 and Resp-1 and between Resp-3 and the controlled process.

Note that in some cases, there might be more than one responsibility that could serve as the feedback source for a piece of required feedback. In such cases, a systems engineer will need to choose which responsibility should serve as the feedback source. For example, in Figure 17, Resp-1 needs to receive feedback FB-5 for PM-5, and either Resp-2 or Resp-3 could provide that feedback because they both have PM-5 in their process model. In Figure 17, Resp-2 is chosen as the feedback source. However, because this behavioral design process is designed to be iterative, this decision can be revisited and changed later if needed.

Next, by applying the control action targets rule, two responsibilities (or a responsibility and the controlled process) will be connected by a control action if the constraint for one responsibility requires that it make use of information contained in a control action provided by another. For example, in Figure 17, Resp-1 provides control action CA-1 to Resp-2 because of the Resp-2 constraint that requires CA-1 be considered in Resp-2's decision making. Similarly, Resp-

2 provides control action CA-3 to the controlled process because of the Resp-2 constraint that requires CA-3 to be used to influence the behavior of the controlled process. This same reasoning is also how the control action targets were determined for the control actions between Resp-1 and Resp-3 and between Resp-3 and the controlled process.

In addition, Figure 17 also illustrates how control inputs (i.e., lateral coordination) between two responsibilities might be defined. If two responsibilities share a common process model part that is not associated with a control action, then lateral coordination or a control input is needed between them to ensure that the shared process model part remains consistent between them and does not become misaligned [95].

As a concrete example of how these rules are applied for a specific system, consider how Resp-1 identified for the simple ATC example might receive the required feedback defined in Figure 16 and what the target of its control action might be. The feedback sources and control action target for the feedback and control actions associated with Resp-1 are shown in Figure 18.

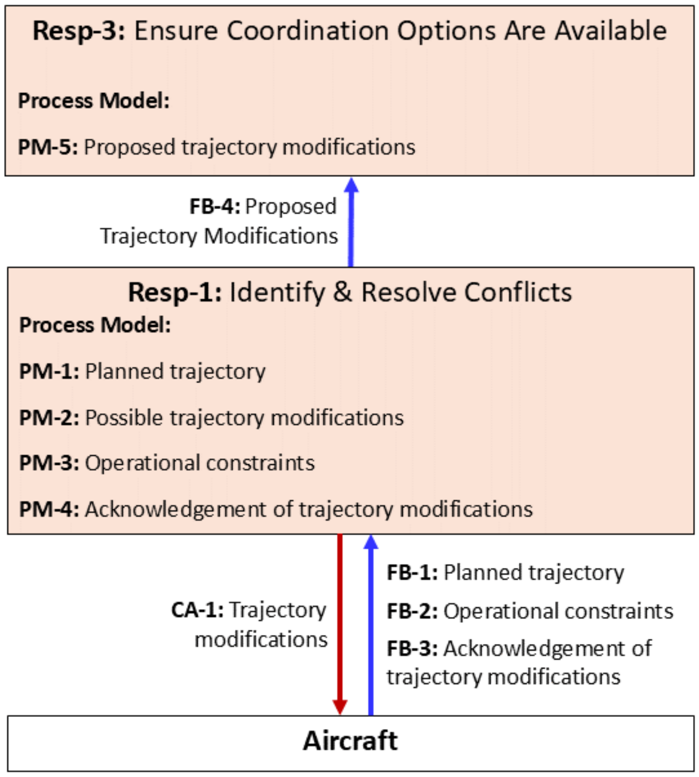

Resp-3: Ensure Coordination Options Are Available
Process Model:
PM-5: Proposed trajectory modifications
FB-4: Proposed Trajectory Modifications
Resp-1: Identify & Resolve Conflicts
Process Model:
PM-1: Planned trajectory
PM-2: Possible trajectory modifications
PM-3: Operational constraints
PM-4: Acknowledgement of trajectory modifications
CA-1: Trajectory modifications
FB-1: Planned trajectory
FB-2: Operational constraints
FB-3: Acknowledgement of trajectory modifications
Aircraft

Figure 18: Identifying control action targets and feedback sources

For Resp-1, FB-1, FB-2, and FB-3 all involve feedback about the aircraft and therefore that feedback is obtained directly from the aircraft. Similarly, CA-1 is intended to change the trajectory of aircraft to resolve a conflict and therefore CA-1 is provided to the aircraft.

Figure 18 also introduces a second responsibility Resp-3 to illustrate how Resp-1 might be a feedback source for another responsibility. Resp-3 is a responsibility that receives proposed trajectory modifications and confirms that an aircraft will always have alternate trajectories available if the proposed trajectory modifications were implemented. Resp-3 therefore needs to know what trajectory modifications are being proposed (PM-5). Since Resp-1 is identifying trajectory modifications to transmit to the aircraft and has possible trajectory modifications in its process model (PM-2), Resp-1 is therefore the source of feedback about proposed trajectory modifications to Resp-3.

Having defined the various control elements and the feedback sources and control action targets, the control elements can now be assembled to create the conceptual architecture. Once assembled, the conceptual architecture looks similar in structure to the control structure in Figure 17. The conceptual architecture therefore represents the desired control behavior of the system in terms of the responsibilities that will need to be performed and the control actions and feedback that are exchanged between them and with the controlled process.

In addition, using the traceability that was maintained throughout this process thus far, each element in the conceptual architecture can be traced directly back to a requirement and an STPA scenario that motivated its inclusion. Thus, this process also helps to capture the design rationale underlying the inclusion of each element in the conceptual architecture.

3.3.3 Updating the Initial STPA and Refining the Conceptual Architecture

Finally, the last step of this behavioral design process is to update the STPA analysis of the system based on the conceptual architecture that was created to refine the causal scenarios identified in the initial SPTA analysis. This STPA update provides a systems engineer with an opportunity to evaluate the conceptual architecture they have created to identify any flaws that might have been inadvertently introduced and the ways in which the conceptual architecture might not adequately control the system hazards identified at the beginning of STPA.

Based on the updated and refined set of STPA scenarios, systems engineers can then decide if any of those scenarios could be mitigated or prevented by changing the conceptual architecture. For example, if a scenario describes a missing piece of feedback, the conceptual architecture should be modified to add the missing feedback. Sometimes, the STPA scenarios may also highlight a problem with how the behavior of a responsibility was designed and that may prompt a reformulation of that responsibility to try to improve its behavior. Once the necessary modifications have been made to the conceptual architecture, the STPA analysis can then be updated again to determine if the change had its desired effect and if any new unsafe behaviors were introduced.

By iteratively updating the conceptual architecture and then updating the STPA analysis, this behavioral design process gives systems engineers the opportunity to iterate on the design of the conceptual architecture and explore the behavioral design space for a system. This iteration is intended to be performed until no further improvements to the conceptual architecture can be made. It is at this point that a system designer can proceed to the structural design process where a system architecture is created to implement this conceptual architecture.

3.4 The Structural Design Process

Once the conceptual architecture is created, the final part of this architecture development framework is to create a system architecture to implement it using the structural design process. The goal of this structural design process is therefore to identify the system architecture that best achieves the desired emergent properties.

To create a system architecture that implements the conceptual architecture, the main design decision that must be made is who is assigned to perform each of the responsibilities.

Once the responsibilities have all been assigned, the control actions and feedback associated with each responsibility can be assigned accordingly to the same controller. This ensures that each controller is provided with the appropriate control actions (i.e., authority) and feedback to perform their assigned responsibilities.

Thus, in this approach, an *architecture option* represents one possible way to assign the responsibilities (and their associated control actions and feedback) to either existing or new controllers in the system. A generic example of how an architecture option is created is shown in Figure 19.

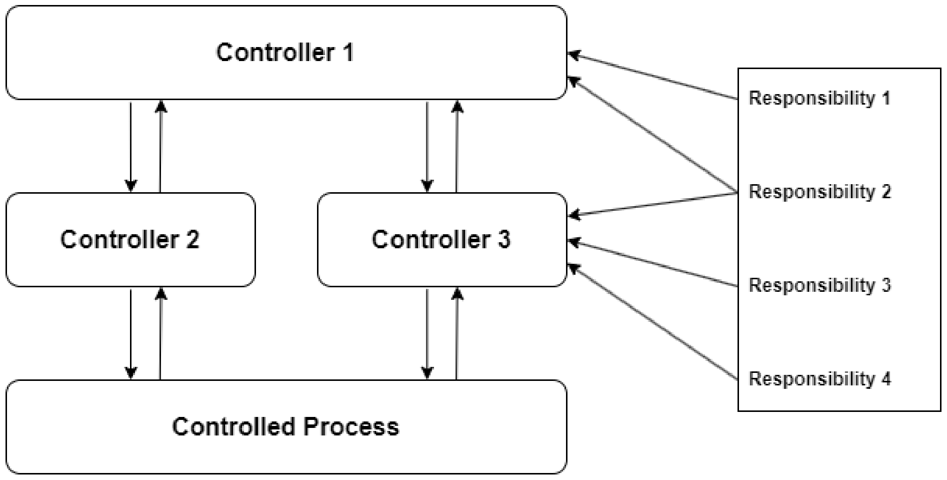


Figure 19: Generic example of how an architecture option is created

The generic system shown in Figure 19 has 3 controllers which collectively control a controlled process and there are four responsibilities that each need to be assigned to at least one controller. Thus, one possible architecture option is:

1. Assign Resp-1 to controller 1
2. Assign Resp-2 to both controller 1 and controller 3 (shared responsibility)
3. Assign Resp-3 and Resp-4 to controller 3

Although Figure 19 illustrates one possible assignment of these responsibilities, it is not the only one. The *tradespace* of possible architecture options is therefore defined by all possible assignments of responsibilities to controllers. However, this tradespace grows exponentially with the number of responsibilities and controllers in the system. In general, for a system with n responsibilities and m possible controller assignments, the number of possible architecture options (i.e., the size of the tradespace) N is defined by Equation 1. Note that this equation assumes each responsibility is only assigned to one controller. Relaxing this assumption and allowing a responsibility to be assigned to multiple controllers further increases the value of N.

$$N = m^n \tag{1}$$

Thus, in the case of the generic example in Figure 19, with n = 4 responsibilities to be assigned and m = 3 possible controllers to assign them to, there are theoretically N = 81 possible architecture options (again, assuming no sharing of responsibilities between controllers). For a real system with many more controllers and responsibilities, there could be an overwhelming number of potential architecture options to consider in the tradespace.

Because the architecture tradespace grows exponentially with the number of responsibilities and controllers in the system, it will not be feasible or practical to exhaustively explore every architecture option before selecting the best or preferred one. Instead of exhaustive enumeration, a process is needed to guide the exploration of different architecture options and to highlight those architectures that are worth exploring and comparing.

To do this, an iterative structural design process was developed to help systems engineers systematically explore alternative architecture options and incrementally improve the system architecture based on what they learn about the behavior of different architecture options. An overview of this process is shown in Figure 20.

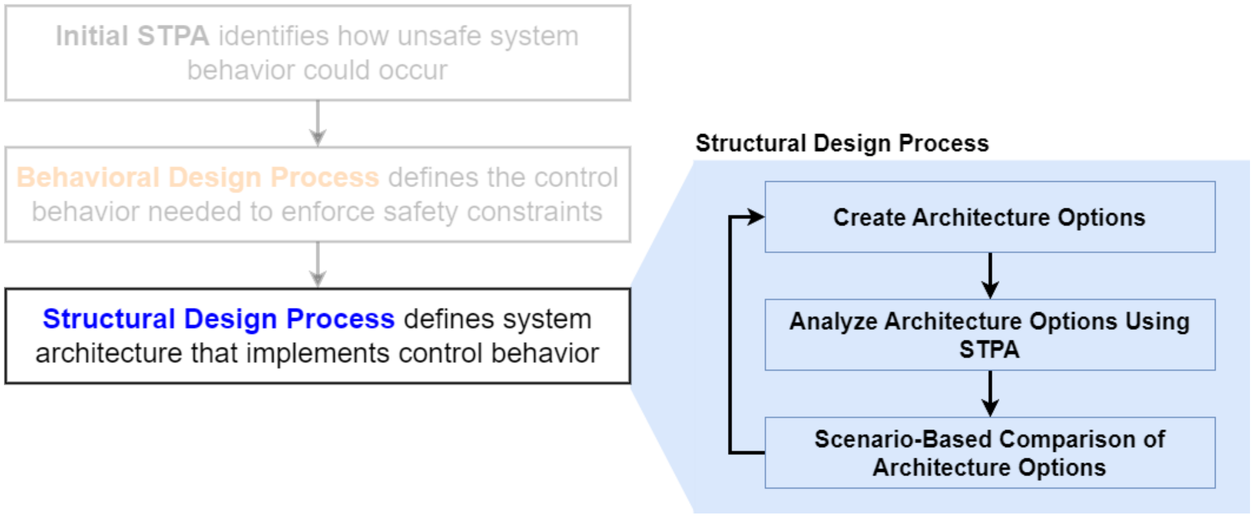

Initial STPA identifies how unsafe system behavior could occur
Behavioral Design Process defines the control behavior needed to enforce safety constraints
Structural Design Process defines system architecture that implements control behavior
Structural Design Process
Create Architecture Options
Analyze Architecture Options Using STPA
Scenario-Based Comparison of Architecture Options

Figure 20: Overview of the structural design process

3.4.1 Creating Architecture Options

The inputs to this process are the conceptual architecture and the causal scenarios from the updated STPA that were identified in the behavioral design process. Recall that in that process, any scenarios identified in the updated STPA that could be mitigated or prevented by making a change to the conceptual architecture were addressed. However, there are also some scenarios that might only be possible to mitigate or prevent with a structural change. It is from these scenarios that architecture options of interest can be identified.

The first step in this structural design process is to use the causal scenarios to identify potential responsibility assignments that could help to mitigate or prevent them. In other words, the goal is to identify what responsibility assignments might be preferable *because* they help to mitigate or prevent unsafe behavior. For example, if a scenario involves two responsibilities having inconsistent information about the same process model part, one way to prevent that scenario occurring could be to assign the responsibilities to the same controller to avoid having the same process model part being needed by two different controllers.

For these scenarios where a preferred responsibility assignment could mitigate or eliminate the occurrence of that scenario, these preferences are recorded as *assignment constraints*. Examples of different types of assignment constraints are shown in Table 3.

Table 3: Examples of different types of assignment constraints

Assignment Constraint Type	**Constraint Notation**
Preferred controller constraint *Note: C_a and C_b are controllers in a system*	Assigning a responsibility to one preferred controller: Resp-X = C_a Shared assignment of a responsibility to multiple controllers: Resp-X = $C_a \wedge C_b$ Multiple assignment preferences for a responsibility: Resp-X = $C_a \vee (C_a \wedge C_b)$
Same Controller Constraint	Resp-X = Resp-Y
Different Controller Constraint	Resp-X $\neq$ Resp-Y

Once these assignment constraints have been identified, they can then be used to decide what architecture options are created and explored. This is done by first creating a baseline architecture option that assigns responsibilities to controllers in a way that satisfies as many of the assignment constraints as possible. Then, for each assignment constraint that is not satisfied by the baseline architecture option, a change is made to the responsibility assignments to satisfy that assignment constraint. Thus, different architecture options are created as changes are made to the assignment of different responsibilities.

3.4.2 Analyzing and Comparing Architecture Options

Once architecture options have been created, the remaining two steps in the structural design process are to analyze and compare the architecture options to understand how different responsibility assignments change the behavior of the architecture. This information can then be used to inform follow-on architectural design decisions. Figure 21 illustrates how architecture options are analyzed and compared.

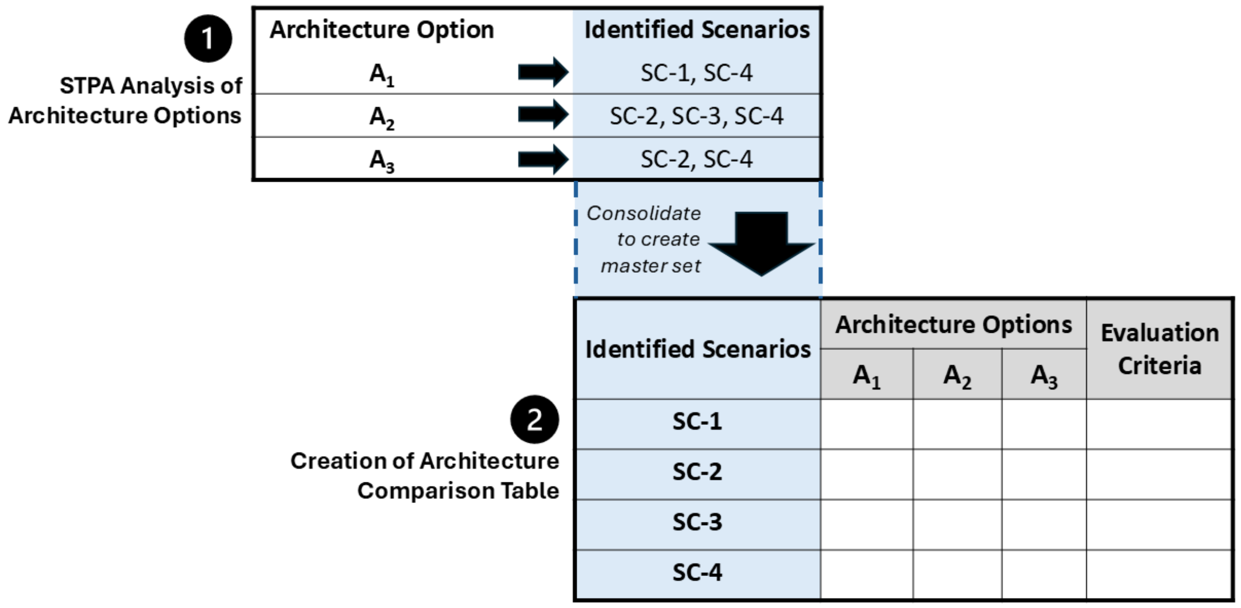


Figure 21: Comparing architecture options based on STPA scenarios

As shown in Figure 21, STPA is first used to analyze each architecture option to determine what unsafe behaviors might occur in each architecture option. Since these architecture options are different implementations of the conceptual architecture, the STPA analysis performed at the end of the behavioral design process can be updated and refined again here to reflect the specific architecture option being analyzed.

From the STPA analysis of each architecture option, causal scenarios are obtained. Some of these causal scenarios might be unique to the given architecture option while others might occur for multiple architecture options. For example, in the upper table of Figure 21, scenario SC-1 is only identified for architecture option A_1 and SC-3 is only identified for A_2. However, SC-2 is identified for both A_2 and A_3.

Regardless of which architecture option the scenario is identified for, all the scenarios identified from the analysis of each architecture option are combined into a master scenario set. This master set is then used to instantiate an *architecture comparison table*. The lower half of Figure 21 shows a generic example of what an architecture comparison table looks like before it is filled out and Table 4 shows what that architecture comparison table looks like once it is completed for the generic example in Figure 21.

Table 4: Generic example of an architecture comparison table once completed

Identified Scenarios	Scenario Occurs?			Evaluation Criteria
	A_1	A_2	A_3	
SC-1	**Yes**	**No** *[Assumption]*	**No** *[Assumption]*	EC-1
SC-2	**No** *[Assumption]*	**Yes**	**Yes**	EC-2
SC-3	**No** *[Assumption]*	**Yes**	**No** *[Assumption]*	EC-3 EC-4
SC-4	**Yes**	**Yes**	**Yes**	N/A

As shown in Table 4, the architecture comparison table has 3 main parts:

1. **Identified scenarios:** One row is created for each scenario in the master scenario set
2. **Scenario occurrence:** One column is created for each architecture option being compared and each cell contains a "yes" or "no" to indicate whether the scenario occurs for that architecture option
3. **Evaluation criteria:** A short phrase describing a control-related difference in behavior between the architecture options

Filling out this architecture comparison table starts with deciding whether each scenario occurs for that architecture option. For example, in the first row of Table 4 scenario SC-1 occurs for architecture option A_1 (because it was identified for that architecture option as shown in the upper table of Figure 21) but is resolved or does not occur for A_2 and A_3. By contrast, in the third row, scenario SC-3 occurs for architecture A_2 but does not occur for A_1 and A_3.

When deciding if a scenario occurs for an architecture option, it is important to record any assumptions used to make that decision, especially if it is decided that a scenario does not occur for an architecture option. In Table 4, these assumptions are denoted by the "*[Assumption]*" placeholders in the cells containing "No". It is important to capture these assumptions because if that architecture option is chosen for further development, the ability of that architecture to resolve or avoid that scenario becomes contingent on those underlying assumptions remaining valid. The supporting framework developed later in Chapter 6 then describes how to ensure they remain valid as the development of the system progresses.

Once these determinations have been made, they can be used to identify the control-related differences between architecture options. For each scenario, the behavior of each architecture option in that scenario is compared and an *Evaluation Criterion* is generated that describes the control-related difference(s) in behavior that differentiates the architecture options. For example, one difference in decision making between centralized and decentralized ATM architectures that has been identified in the literature [96] could be described by the following evaluation criterion.

Example evaluation criterion: Responsiveness of trajectory modification decisions to prevent loss of separation when resolving a multi-aircraft conflict in densely populated airspace

By doing this for each of the scenarios in the master set, the evaluation criteria that are generated highlight the various control-related differences between the architecture options.

When generating the evaluation criteria, note that each scenario does not necessarily generate a unique evaluation criterion and it is possible that multiple evaluation criteria might be identified for a given scenario. This might occur if there is more than 1 aspect of the scenario where differences in behavior are observed between the architecture options. For example, in the third row of Table 4, criteria EC-3 and EC-4 are both derived from the same scenario SC-3.

Note also that not every scenario will have an evaluation criterion generated for it because there may be some scenarios that are found to occur for all architecture options. Scenario SC-4 (the last row of Table 4) is an example of this. This result would suggest that the unsafe behavior described in that scenario is not prevented or mitigated by any of the architecture options and therefore no meaningful control-related difference in behavior is observed between architecture options for that scenario. Thus, as shown in Table 4, no evaluation criterion is generated.

To help guide an analyst or systems engineer in generating an evaluation criterion from a given scenario, Figure 22 shows the general structure of an evaluation criterion.

Evaluation Criterion Structure:

Example: Responsiveness of trajectory modification decisions to prevent loss of separation when resolving a multi-aircraft conflict in densely populated airspace

<Characteristic> of **<Control Aspect>** to prevent **<hazard>** when **<scenario context>**

[1] *[2]* *[3]* *[4]*

Figure 22: Structure of an evaluation criterion

As shown in Figure 22, an evaluation criterion consists of four parts, each of which provides control-relevant information to support the comparison of architecture options. The first two parts (items 1 and 2) describe what is different about the control behavior of the architecture options in the scenario under consideration. Consistent with the concept of control in systems theory and STAMP, there are four *Control Aspects* that should be considered: (1) decision making, (2) process models, (3) feedback and control inputs, and (4) control path. The *Characteristic* then describes a property or attribute of that aspect of control. Table 5 provides some example characteristics for each control aspect. Note that the characteristics listed in Table 5 are intended to be used as examples only and do not represent an exhaustive list of every possible attribute or property that could be identified.

Table 5: Example characteristics for each control aspect

Control Aspect	**Example Characteristics**
Decision Making	• Responsiveness of decision making • Frequency or complexity of decision making • The need/ability to make a decision (in certain situations) • Ease of coordinating two related decisions
Process Models	• Level of situational awareness available or needed • Ability to ensure adequate update of a process model part • Level of uncertainty associated with a process model part • Ability to maintain alignment of two related process model parts or the same process model part across two controllers
Feedback and Control Inputs	• Timeliness of feedback or control input • Ability to interpret/process/verify/respond appropriately to feedback or a control input • Use of a certain type of feedback or control input
Control Path	• Vulnerability of a control path or control action • Potential for conflict between two related control actions or the same control action issued by two different controllers • Responsiveness of controlled process in executing control action

The latter two parts of an evaluation criterion (items 3 and 4) describe the conditions under which the control behavior described in the first two parts occurs. This includes the *Hazard* that the control behavior is intended to prevent and the *Scenario Context* in which that control behavior is occurring. Combined, these two parts describe why the control behavior is needed and the circumstances in which it is occurring and both are derived from the causal scenario under consideration. The hazard is derived from the traceability maintained in STPA between the scenario and the system-level hazards and the scenario context is derived from the scenario itself.

Once the architecture comparison table has been completed, the evaluation criteria and the results in the comparison table can be used to identify benefits and tradeoffs between the architecture options. For each evaluation criteria, if a scenario does not occur for that architecture option, there is a benefit for that architecture option with respect to that evaluation criterion (e.g., better, more responsive, more timely). However, if a scenario does occur for that architecture option, then there is a tradeoff for that architecture option with respect to that evaluation criterion. For example, Table 6 shows how the comparison table in Table 4 would be used to generate benefits and tradeoffs for the three generic architecture options.

Table 6: Comparison results for generic evaluation criteria in Table 4

Evaluation Criteria	**Benefit (+) or Tradeoff (-)**		
	A_1	**A_2**	**A_3**
EC-1	⊖	⊕	⊕
EC-2	⊕	⊖	⊖
EC-3	⊕	⊖	⊕
EC-4	⊕	⊖	⊕

By analyzing the comparison results like this, a systems engineer can study the different control-related aspects of a system's behavior more systematically to both identify the benefits and tradeoffs of different architecture options and understand what parts of the architecture contributed to those benefits or tradeoffs.

Once the comparison results have been generated, systems engineers have two options for how to make use of these results. One option is that they could decide that one of the architecture options being compared is the best architecture they can find and therefore they choose one of those options as the system architecture to move forward with for further development. Alternatively, they could decide that there exists one or more additional architecture options that might be better than the ones that have already been identified. For example, an architecture option representing a combination or hybrid of the responsibility assignments in two of the already-compared architecture options might be considered. If this is the case, they can continue iterating through this structural design process by creating those additional architecture options and then analyzing and comparing those new options using this same process until they believe they have found the best or their preferred system architecture to move forward with for further development.

3.5 Summary

This chapter introduced the safety-driven architecture development framework that was developed to enable systems engineers to design safety and other desired emergent properties into their system architectures from the beginning of development. Unlike existing approaches to architecture development, this new approach does not rely on decomposition to create the system architecture. Instead, it is based on systems theory and focuses on helping systems engineers to analyze and design the control-oriented aspects of the system to ensure that appropriate controls are implemented in the system architecture. This ensures that the system can adequately enforce the necessary safety constraints to avoid unsafe or undesirable system behavior.

The key idea behind this safety-driven architecture development framework is to use STPA results to inform behavioral and structural design decisions and there are three main parts to the framework. First, an initial STPA analysis of the system identifies preliminary information about how unsafe behavior could occur. Then, the *Behavioral Design Process* provides a structured way to define the desired control behavior of the system modeled as a conceptual architecture. Finally, the *Structural Design Process* provides a systematic way to explore and compare different system architecture options for implementing the conceptual architecture. Putting all three parts together, the full safety-driven architecture development framework is shown in Figure 23.

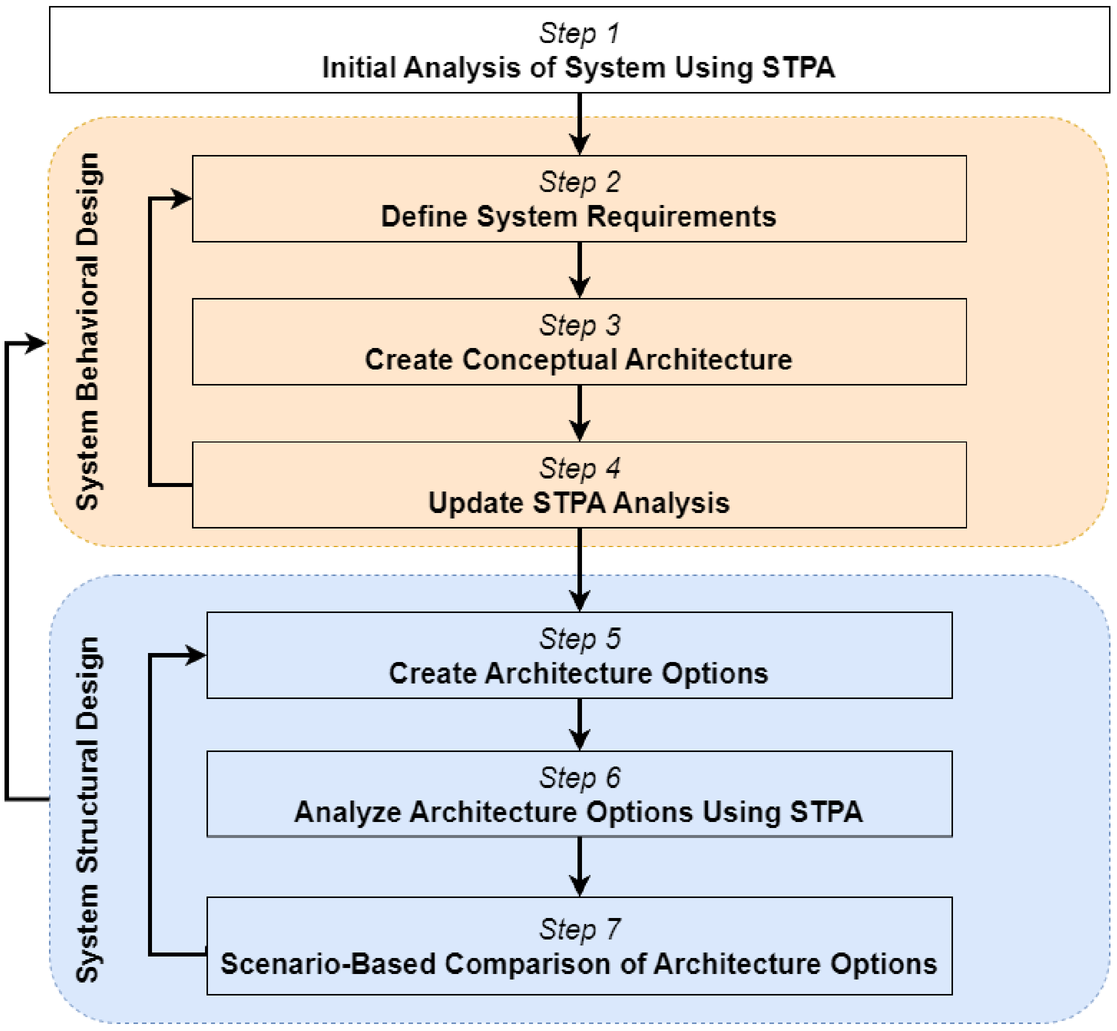

Step 1
Initial Analysis of System Using STPA
System Behavioral Design
Step 2
Define System Requirements
Step 3
Create Conceptual Architecture
Step 4
Update STPA Analysis
System Structural Design
Step 5
Create Architecture Options
Step 6
Analyze Architecture Options Using STPA
Step 7
Scenario-Based Comparison of Architecture Options

Figure 23: The full safety-driven architecture development framework

In the next two chapters, this safety-driven architecture development framework is applied to develop an ATM system architecture for the NAS that will enable the integration of UAM into the airspace alongside existing air traffic.

Chapter 4 Design Iteration 1: Developing an Initial ATM Architecture

Since the 1930s, commercial/civil air traffic in the NAS has been managed using a centralized ATM architecture [24] where Air Traffic Control (ATC) is primarily responsible for keeping aircraft safely separated, especially those flying under Instrument Flight Rules (IFR). Although this centralized architecture has enabled a safe NAS thus far, it will be challenging to continue relying on it while introducing UAM. This is because, as discussed in Chapter 1, the characteristics of UAM air traffic are challenging conventional approaches used to ensure the safety of the NAS.

Because of these challenges, a significant amount of research has been done to define potential new ATM concepts and architectures that could feasibly manage UAM air traffic. These include more decentralized ATM concepts such as Free Flight [97], Distributed Air/Ground Traffic Management (DAG-TM) [98], and more automated approaches to ATM [66, 67, 99]. NASA and the FAA have also both published concept of operations documents for UAM [17, 100] that describes the infrastructure that will be needed and the airspace structure that might be used to safely manage UAM air traffic.

Despite the wide variety of new ATM architectures that have been proposed, the methods used to analyze or evaluate these architectures for safety face the same challenges described in Section 2.2. Often, the safety of these ATM architectures is evaluated only after they have been created. In addition, it can be difficult to evaluate the safety of these ATM architectures using quantitative metrics, especially early in the development process.

For this reason, the goal of this first design iteration is to apply the architecture development framework developed in Chapter 3 to develop an ATM architecture for UAM that accounts for safety considerations from the beginning. This will be done by identifying and then comparing different possible architecture options to inform a decision about what the preferred ATM architecture for UAM should be.

Because the ATM architectures developed in the existing literature have primarily focused on collision avoidance, this design iteration will focus on the same. By aligning the focus of this first design iteration with that of previous comparisons of ATM architectures, the results obtained from this case study can be compared to those in the existing literature to evaluate the ability of this framework to identify suitable criteria for comparing architecture options.

The remainder of this chapter is organized as follows. First, an initial STPA of the NAS is performed to determine how unsafe behavior might occur when UAM air traffic is introduced into the airspace. Then, NAS system requirements for collision avoidance and a conceptual architecture that meets those requirements is developed. Finally, two architecture options to implement the conceptual architecture are evaluated and compared to determine the benefits and tradeoffs between them. These benefits and tradeoffs are then compared to those identified in the existing literature. Finally, the benefits and tradeoffs are used to inform a decision about what the preferred ATM architecture for UAM should be.

4.1 Initial Analysis of the NAS Using STPA

STPA begins with the identification of relevant losses, hazards, and system-level safety constraints. Because this case study focuses on the safe management of air traffic, the system boundary matches that of the NAS today and includes the various aircraft and operators, the people and components needed to manage air traffic, and the FAA. The losses and hazards also correspond to those of the NAS and are presented in Table 7 and Table 8 respectively. The system-level safety constraints derived from those hazards are presented in Table 9.

Table 7: System losses

Loss ID	Loss Description
L-1.	Loss of life or injury
L-2.	Loss or damage to aircraft or equipment
L-3.	Nonachievement of mission
L-4.	Excessive environmental impact (beyond <TBD> level)
L-5.	Loss or damage of critical infrastructure
L-6.	Loss of critical community needs
L-7.	Loss of public acceptance of UAM

Table 8: System hazards

Hazard ID	Hazard Description	Loss Link
H-1.	Aircraft do not maintain minimum separation (to other flights or surface objects)	L-1, L-2, L-3, L-5, L-7
H-2.	Flight operations are harmful to occupant health	L-1, L-3, L-7
H-3.	Missions (e.g., transportation, police operations) cannot be completed within acceptable performance limits (e.g., within a specified period of time, within delay tolerance)	L-3, L-4, L-7
H-4.	Environmental effects of flight operations exceed acceptable levels (e.g., noise, emissions)	L-4, L-7
H-5.	Critical public or aviation infrastructure becomes inoperable	L-2, L-3, L-5, L-6
H-6.	Public safety is compromised (e.g., because emergency services aircraft are unable to fulfill their mission or airspace exclusions are not maintained)	L-1, L-2, L-3, L-5, L-6, L-7

Table 9: System-level safety constraints

Constraint ID	Constraint Description	Hazard Link
C-1.	Aircraft must not violate minimum separation standards in flight (to air and surface objects)	H-1
C-2.	Flight operations must not be harmful to occupant health	H-2
C-3.	Missions must be completed within acceptable performance limits	H-3
C-4.	Environmental effects of flight operations must not exceed acceptable levels (e.g., noise, pollution)	H-4
C-5.	Critical infrastructure must remain operable	H-5
C-6.	Flight operations must not compromise public safety	H-6

Once the losses, hazards, and safety constraints have been identified, the next step in STPA is to create the control structure. The control structure used to model the NAS is shown in Figure 24. To minimize the number of assumptions needed to create this initial control structure, the NAS is modeled at a high level of abstraction and this control structure will be incrementally refined as architecture development progresses.

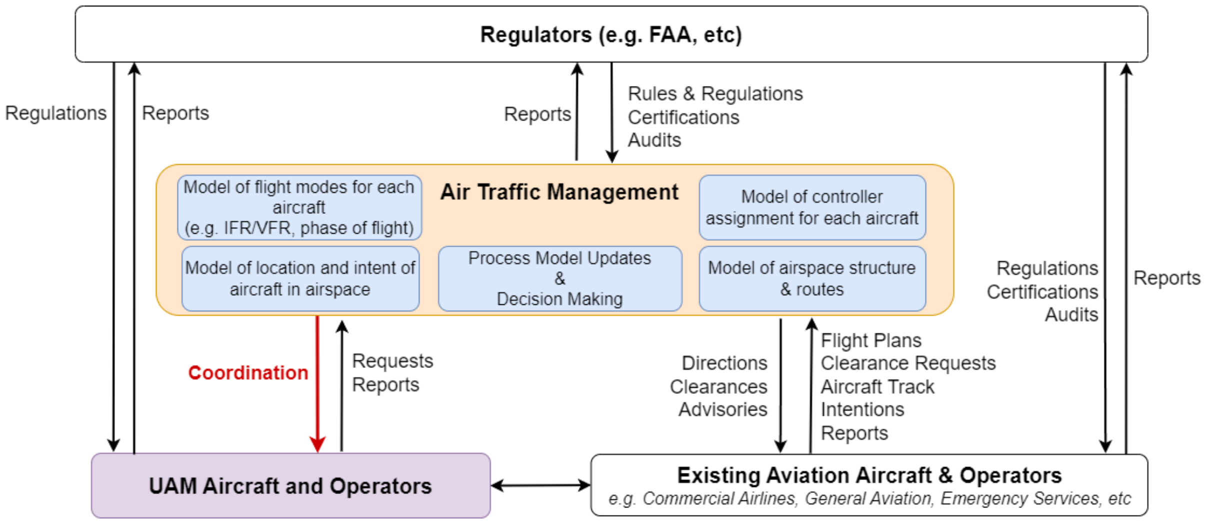

Regulators (e.g. FAA, etc)
Regulations
Reports
Reports
Rules & Regulations
Certifications
Audits
Air Traffic Management
Model of flight modes for each aircraft (e.g. IFR/VFR, phase of flight)
Model of controller assignment for each aircraft
Model of location and intent of aircraft in airspace
Process Model Updates & Decision Making
Model of airspace structure & routes
Regulations
Certifications
Audits
Reports
Coordination
Requests
Reports
Directions
Clearances
Advisories
Flight Plans
Clearance Requests
Aircraft Track
Intentions
Reports
UAM Aircraft and Operators
Existing Aviation Aircraft & Operators
e.g. Commercial Airlines, General Aviation, Emergency Services, etc

Figure 24: NAS control structure

Since the goal is to analyze the NAS with UAM integrated into it, the lowest level of the control structure includes both UAM aircraft and operators as well as existing aviation aircraft and operators, including commercial airlines, general aviation (GA) aircraft, and emergency services. The next level up in the control structure is Air Traffic Management, an abstract controller that encapsulates all ATM responsibilities necessary to safely manage and control both existing aviation air traffic as well as UAM air traffic. Note that this abstract controller does not imply that a decision has been made about whether UAM air traffic is managed by the same entity as existing aviation air traffic or a separate one. That decision should be made as part of developing the ATM architecture for the NAS. This abstract controller is simply a model abstraction used to enable a broader analysis of air traffic management in the NAS. Finally, the highest level of the control structure includes federal regulators such as the Federal Aviation Administration (FAA).

It is worth noting that the control actions and feedback associated with UAM aircraft and operators (left side of Figure 24) are modeled more abstractly than those associated with existing aviation operations (right side of Figure 24). This reflects what is already known or commonly assumed about how the NAS might accommodate the introduction of UAM. It is commonly assumed that the management of existing air traffic will remain similar to how ATC works today [18]. Therefore, the interactions between ATM and existing aviation aircraft and operators are modeled to reflect those interactions today. However, because the interactions between ATM and UAM aircraft have yet to be determined, no assumption about existing operations or a pre-existing operational concept is made. 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 development process, these abstract control actions and feedback will be refined into more detailed ones based on the desired ATM behavior generated using this framework.

The third step in STPA is to generate UCAs. Since the focus of this research is on designing the ATM system to manage UAM air traffic, the *Coordination* control action highlighted in red in Figure 24 was analyzed to identify UCAs and scenarios. It is well recognized that the NAS will need to exhibit not only safety but also other emergent properties such as throughput and efficiency. Thus, the UCAs (and scenarios) generated during this initial STPA demonstrate how multiple emergent properties can be analyzed in an integrated manner. A selected set of example UCAs involving safety, throughput, and efficiency are presented here to illustrate how these properties were considered. UCA-1.15 and UCA-1.28 also illustrate how UCAs can affect multiple properties (e.g., safety and throughput). The full set of UCAs and scenarios can be found in Appendix A.

<u>Examples of UCAs Involving Safety Concerns</u>

UCA-1.1: Air Traffic Management does not coordinate the interaction between two UAM aircraft or a UAM aircraft and another airspace user when a collision between them is imminent [H-1, H-3]

UCA-1.2: Air Traffic Management does not coordinate air traffic in the airspace to assist UAM aircraft in an emergency [H-1, H-2, H-3]

UCA-1.29: Air Traffic Management coordinates the interaction between two aircraft too late to prevent violation of minimum separation between them [H-1, H-2, H-3]

<u>Examples of UCAs Involving Efficiency Concerns</u>

UCA-1.4: Air Traffic Management does not coordinate air traffic to allow UAM aircraft to access the airspace when UAM aircraft need to execute a mission and the UAM aircraft meet the criteria for access to that airspace [H-3]

UCA-1.8: Air Traffic Management does not coordinate the movements of UAM aircraft when they interfere with the operations of other NAS users [H-1, H-3]

UCA-1.31: Air Traffic Management coordinates air traffic to allow UAM aircraft access to the airspace too late after the time window in which UAM aircraft need that access [H-3]

<u>Examples of UCAs Involving Throughput Concerns</u>

UCA-1.15: Air Traffic Management coordinates air traffic to allow UAM aircraft to access the airspace when the NAS does not have sufficient capacity [H-1, H-3, H-4]

UCA-1.28: Air Traffic Management provides coordination to UAM aircraft that does not satisfy priority needs (e.g., an aircraft running out of fuel needs access to an airport sooner than one that has plenty of fuel) [H-1, H-2, H-3]

UCA-1.39: Air Traffic Management restricts air traffic for too long after environmental effects of system operation have returned to acceptable levels [H-3]

The last step of STPA is to generate causal scenarios for each of the UCAs and several example causal scenarios are presented here for UCA-1.1 and UCA-1.8. These example scenarios illustrate that by starting the STPA analysis early in the development process at a high level of abstraction, a wide variety of different types of scenarios can be identified.

Example Scenarios for UCA-1.1

UCA-1.1: Air Traffic Management does not coordinate the interaction between two aircraft when a collision between them is imminent [H-1, H-3]

CS-1.1.1-2: Air Traffic Management has received feedback about the potential conflict but does not issue coordination because it is preoccupied with other tasks and does not have the capacity to process the feedback it receives. Air Traffic Management therefore does not recognize the potential conflict and does not provide coordination to prevent it.

CS-1.1.2-2: Air Traffic Management does not receive feedback about the potential conflict because there are more aircraft in the airspace than Air Traffic Management is capable of detecting and tracking simultaneously. As a result, it receives incomplete feedback about the aircraft present in the airspace.

CS-1.1.4-1.2: Air Traffic Management provides coordination and it is received by the aircraft but is not effective in preventing violation of minimum separation. This might occur if the aircraft is preoccupied with another task and is unable to execute the coordination provided by Air Traffic Management in a timely manner. It may also occur if the provided coordination is incorrect or insufficient for resolving the conflict.

Example Scenarios for UCA-1.8

UCA-1.8: Air Traffic Management does not coordinate the movements of UAM aircraft when they interfere with the operations of other NAS users [H-1, H-3]

CS-1.8.1-2: Although Air Traffic Management receives feedback about this interference, it does not issue coordination because Air Traffic Management wrongly believes that UAM aircraft's impact on other NAS users is negligible or tolerable by the other NAS users and therefore there is no need to issue coordination to reduce the impact.

CS-1.8.2-4: Air Traffic Management does not receive feedback that UAM aircraft are interfering with the operations of other NAS users because the impact to their operations occurs gradually or there is a small impact to a large number of NAS users and Air Traffic Management does not receive feedback about the overall extent of the impact to the operations of other NAS users.

CS-1.8.4-2: Air Traffic Management provides coordination when UAM aircraft interfere with the operations of other NAS users. The coordinated solution is received by the UAM aircraft but it does not prevent the interference because it was provided by Air Traffic Management at the last minute.

4.2 Developing the Collision Avoidance Conceptual Architecture

The next part of this architecture development framework is the behavioral design process where a conceptual architecture is developed to adequately control the hazards and prevent undesirable behavior. As discussed at the beginning of this chapter, this design iteration is primarily focused on safety and the collision avoidance aspect of air traffic management. Thus, a conceptual architecture for collision avoidance was developed to mitigate or prevent the UCAs and scenarios that will lead to H-1 (i.e., violation of minimum separation). Although not demonstrated in this research, this same process can be applied to control the other hazards.

4.2.1 Identifying NAS System Requirements for Collision Avoidance

The behavioral design process starts with identifying the system requirements that describe the safety constraints necessary to mitigate or prevent the scenarios identified using STPA from occurring. Figure 25 shows examples of how system requirements were derived from specific UCAs and scenarios in the initial STPA analysis. Traceability between scenarios and requirements is recorded using the links and "↓" symbol in the square braces. This traceability records the rationale for each requirement by linking it to the scenario that each requirement is intended to mitigate or prevent.

Table 10 then shows some additional examples of collision avoidance requirements that were generated. The full set of collision avoidance requirements generated for this design iteration is presented in Appendix B.

UCA-1.1: Air Traffic Management does not coordinate the interaction between two UAM aircraft or a UAM aircraft and another airspace user when a collision between them is imminent *[H-1, H-3]*
CS-1.1.1-2: Air Traffic Management has received feedback about the potential conflict but does not issue coordination because it is preoccupied with other tasks and does not have the capacity to process the feedback it receives. Air Traffic Management therefore does not recognize the potential conflict and does not provide coordination to prevent it. [↓ Req-3, Req-4] **CS-1.1.2-2:** Air Traffic Management does not receive feedback about the potential conflict because there are more aircraft in the airspace than Air Traffic Management is capable of detecting and tracking simultaneously. As a result, it receives incomplete feedback about the aircraft present in the airspace. [↓ Req-8]
Req-3: ATM system shall ensure that sufficient capacity is available to detect and coordinate all aircraft that have or will need access to the airspace *[CS-1.1.1-2]* **Req-4:** ATM system shall coordinate the movement of aircraft to resolve any potential conflicts *[CS-1.1.1-2]* **Req-8:** ATM system shall only allow as many users to access the airspace as it is capable of detecting, tracking and coordinating *[CS-1.1.2-2]*

Figure 25: Examples of how solution-neutral, system-level requirements are generated

Table 10: Additional examples of system requirements

Req ID	Requirement
Req-6	ATM system shall ensure that acceptable coordination options are always available for aircraft to avoid violation of minimum separation.
Req-10	ATM system shall account for intended movements of aircraft in addition to current trajectories to detect potential collisions
Req-11	ATM system shall ensure that information about the intent, mission, acceptable operational impacts and future intended movements of aircraft is available, does not contain errors and is kept updated
Req-12	ATM system shall coordinate the movements of other aircraft to prevent violation of minimum separation with an aircraft that is unable to communicate or not responding
Req-13	ATM system shall ensure that aircraft have acknowledged receipt of the coordination being communicated
Req-17	ATM system shall ensure that coordination provided to the aircraft does not cause additional violation of minimum separation
Req-83	ATM system shall ensure that any proposed coordination has new alternative trajectories available before issuing the proposed coordination

4.2.2 Creating the Conceptual Architecture

Once the system requirements have been generated, the next step in the behavioral design process is to create a conceptual architecture to define the control behavior that will ensure UAM air traffic is safely managed. As described in Chapter 3, this is done by first categorizing the system requirements as either control requirements or constraint requirements. As an example, Table 11 shows how the ten requirements shown in Figure 25 and Table 10 are categorized.

Table 11: Example categorization of control requirement and constraint requirements

Category	Requirement
Control Requirements	**Req-3:** ATM system shall ensure that sufficient capacity is available to detect and coordinate all aircraft that have or will need access to the airspace
	Req-4: ATM system shall coordinate the movement of aircraft to resolve any potential conflicts
	Req-6: ATM system shall ensure that acceptable coordination options are always available for aircraft to avoid violation of minimum separation.
	Req-8: ATM system shall only allow as many users to access the airspace as it is capable of detecting, tracking and coordinating
	Req-11: ATM system shall ensure that information about the intent, mission, acceptable operational impacts and future intended movements of aircraft is available, does not contain errors and is kept updated
Constraint Requirements	**Req-10:** ATM system shall account for intended movements of aircraft in addition to current trajectories to detect potential collisions
	Req-12: ATM system shall coordinate the movements of other aircraft to prevent violation of minimum separation with an aircraft that is unable to communicate or not responding
	Req-13: ATM system shall ensure that aircraft have received the coordination being communicated
	Req-17: ATM system shall ensure that coordination provided to the aircraft does not cause another violation of minimum separation
	Req-83: ATM system shall ensure that any proposed coordination has new alternative trajectories available before issuing the proposed coordination

Of the ten requirements shown in Table 11, the first five requirements (blue rows) are control requirements because they describe specific control functions or decisions that need to be made. By contrast, the latter five requirements (green rows) describe constraints or specifications for how conflicts should be resolved.

Once the requirements have been categorized as control or constraint requirements, groups of requirements can then be created where each group is defined by a control requirement and the related constraint requirements that apply to it. A control responsibility and associated responsibility constraints can then be generated for each group. As an example, Req-4 describes the need to prevent conflicts and Req-10, Req-12, Req-13, Req-17, and Req-83 all describe restrictions on how conflicts should be resolved. Thus, these six related requirements can be grouped together.

Table 12 shows the control responsibility and associated constraints that are derived from these requirements, and each responsibility and constraint is traced to the requirement it was derived from using the links in the square braces. The four other control responsibilities identified in this design iteration are shown in Table 13 and were derived from the other four control requirements listed in Table 11.

Table 12: Example derivation of control responsibility and associated constraints

<u>Requirements Group</u>
Req-4: ATM system shall coordinate the movement of aircraft to resolve any potential conflicts
Req-10: ATM system shall account for intended movements of aircraft in addition to current trajectories to detect potential collisions
Req-12: ATM system shall coordinate the movements of other aircraft to prevent violation of minimum separation with an aircraft that is unable to communicate or not responding
Req-13: ATM system shall ensure that aircraft have received the coordination being communicated
Req-17: ATM system shall ensure that coordination provided to the aircraft does not cause another violation of minimum separation
Req-83: ATM system shall ensure that any proposed coordination has new alternative trajectories available before issuing the proposed coordination
<u>Control Responsibility (Resp) and Associated Constraints (RC)</u>
Resp-1: Coordinate the movement of aircraft to prevent conflicts *[Req-4]*
RC-2: Account for planned trajectory when identifying conflicts *[Req-10]*
RC-4: Ensure coordination decisions do not cause secondary conflicts *[Req-17]*
RC-15: Continue resolving conflicts even if one or more aircraft are unable to communicate or are not responding *[Req-12]*
RC-26: Ensure that aircraft have received coordination being communicated *[Req-13]*
RC-58: Confirm alternative trajectories are available for any proposed coordination *[Req-83]*

Table 13: The four other control responsibilities for collision avoidance

Control Requirement	Control Responsibility
Req-3: ATM system shall ensure that the number of active flights in the airspace does not exceed its capacity to detect and coordinate any eminent collisions between any aircraft in the airspace	**Resp-2:** Ensure sufficient capacity is available
Req-6: ATM system shall ensure that acceptable coordination options are always available for aircraft to avoid violation of minimum separation.	**Resp-3:** Ensure coordination options are available
Req-8: ATM system shall only allow as many users to access the airspace as it is capable of detecting, tracking and coordinating.	**Resp-4:** Manage access to the airspace
Req-11: ATM system shall ensure that information about the intent, mission, acceptable operational	**Resp-5:** Manage airspace state information

impacts and future intended movements of aircraft is available, does not contain errors, and is kept updated.	

Having defined these five responsibilities and their associated constraints, the required process model parts, control actions, and feedback can then be identified. Table 14 shows an example of the control elements defined for Resp-1. Table 15 shows an example of the control elements defined for Resp-3. The full set of control actions and feedback for all five responsibilities in this design iteration are shown in Appendix B.

For each of the feedback and control actions in Table 14 and Table 15, the feedback sources and control action targets are also identified using the process described in Section 3.4. In addition, traceability between each control element and the responsibility or associated constraint that was used to generate it is recorded using the links in the square braces.

Table 14: Identifying process model parts, control actions, and feedback for Resp-1

Resp-1: Coordinate the movement of aircraft to prevent conflicts **RC-2:** Account for planned trajectory when identifying conflicts **RC-4:** Ensure coordination decisions do not cause secondary conflicts **RC-15:** Continue resolving conflicts even if one or more aircraft are unable to communicate or are not responding **RC-26:** Ensure that aircraft have received the coordination being communicated **RC-58:** Confirm alternative trajectories are available for any proposed coordination	
Summary of Desired Behavior	If any object or aircraft is within <TBD distance> of any aircraft in the airspace with a closure rate of <TBD closure rate>, a collision is imminent and the two aircraft should be provided with direction to avoid a potential collision. If an aircraft is unable to communicate, the trajectories of other aircraft should be modified to avoid conflicting with the non-communicative aircraft. *<Rationale for the threshold distance and closure rate >*
Process Model Parts & Required Feedback/Inputs	• <u>Feedback from the aircraft:</u> o Aircraft track (position, heading, ID, speed) *[Resp-1]* o Planned trajectory *[RC-2, RC-4]* o Acknowledgement of trajectory modifications *[RC-26]* • <u>Input from Resp-5:</u> Aircraft not communicating *[RC-15]* • <u>Input from Resp-3:</u> Alternate trajectories available *[RC-58]*
Required Control Actions/Outputs	• <u>Control action to the aircraft and output to Resp-5:</u> Trajectory modifications *[Resp-1]* • <u>Control action to the aircraft only:</u> Request acknowledgement of trajectory modifications *[RC-26]*

Table 15: Identifying process model parts, control actions, and feedback for Resp-3

Resp-3: Ensure coordination options are available **RC-25:** Ensure that there is sufficient airspace available to allow alternative trajectories to be selected **RC-54:** Ensure that initiated traffic management plans are accounted for when identifying coordination options **RC-88:** Ensure that further coordination options are available for any proposed coordination	
Summary of Desired Behavior	Continuously evaluate the trajectory and track of all aircraft to ensure that there are always alternative movement options available for the aircraft if its trajectory needs to be modified. If an aircraft's trajectory is being modified, the proposed new trajectory should be evaluated to ensure it has alternative movement options available.
Process Model Parts & Required Feedback/Inputs	• <u>Feedback from the aircraft:</u> o Aircraft track (position, heading, ID, speed) *[Resp-3]* o Planned trajectory *[Resp-3]* • <u>Input from Resp-2:</u> Active traffic management programs *[RC-54]* • <u>Feedback from Resp-1:</u> Proposed trajectory modifications *[RC-88]*
Required Control Actions/Outputs	• <u>Control action to Resp-4 and Resp-1:</u> Alternate trajectories *[Resp-3, RC-25]* • <u>Control action to Resp-1:</u> Confirmation/rejection of trajectory modifications *[RC-88]*

Having defined the five control responsibilities and their corresponding control actions and feedback, an initial conceptual architecture for collision avoidance can be created and this is shown in Figure 26.

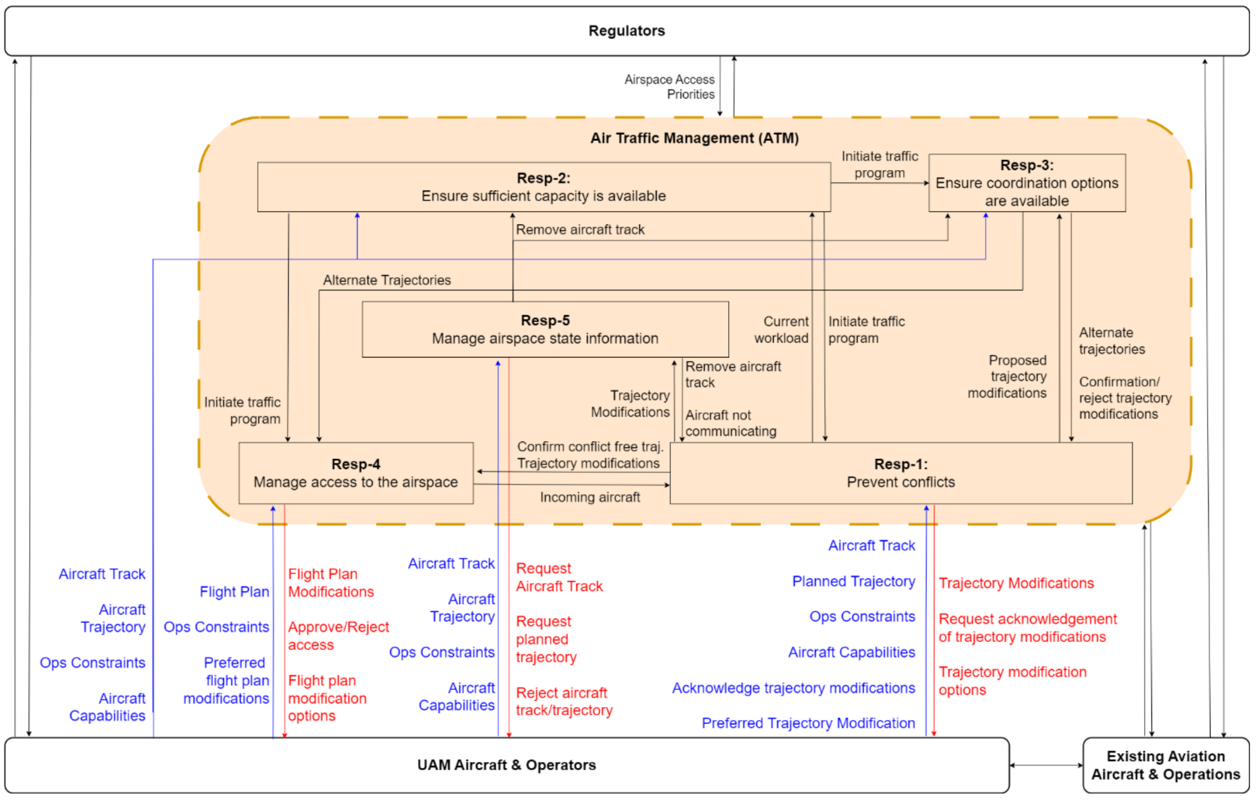

Regulators
Airspace Access Priorities
Air Traffic Management (ATM)
Resp-2: Ensure sufficient capacity is available
Initiate traffic program
Resp-3: Ensure coordination options are available
Remove aircraft track
Alternate Trajectories
Resp-5 Manage airspace state information
Current workload
Initiate traffic program
Alternate trajectories
Proposed trajectory modifications
Confirmation/ reject trajectory modifications
Remove aircraft track
Trajectory Modifications
Aircraft not communicating
Initiate traffic program
Confirm conflict free traj. Trajectory modifications
Resp-4 Manage access to the airspace
Resp-1: Prevent conflicts
Incoming aircraft
Aircraft Track
Aircraft Trajectory
Ops Constraints
Aircraft Capabilities
Flight Plan
Ops Constraints
Preferred flight plan modifications
Flight Plan Modifications
Approve/Reject access
Flight plan modification options
Aircraft Track
Aircraft Trajectory
Ops Constraints
Aircraft Capabilities
Request Aircraft Track
Request planned trajectory
Reject aircraft track/trajectory
Aircraft Track
Planned Trajectory
Ops Constraints
Aircraft Capabilities
Acknowledge trajectory modifications
Preferred Trajectory Modification
Trajectory Modifications
Request acknowledgement of trajectory modifications
Trajectory modification options
UAM Aircraft & Operators
Existing Aviation Aircraft & Operations

Figure 26: Initial conceptual architecture

As shown in Figure 26, this conceptual architecture now provides a more detailed definition of what needs to be contained in the ATM architecture to safely manage UAM air traffic. The five control responsibilities listed in Table 13 refine the orange "Air Traffic Management" box and specify the control actions and feedback needed to safely manage air traffic. Starting at the bottom of the orange box, the first row of responsibilities are Resp-1 and Resp-4. Resp-1 is the responsibility for identifying and resolving conflicts, and Resp-4 is the responsibility for managing access to the airspace and ensuring that aircraft only enter UAM airspace when they meet the requirements for operating in it.

Above these responsibilities is Resp-5, the responsibility for managing information about the state of the airspace. This includes ensuring that aircraft track and trajectory data is not tampered with and that any erroneous track data for an aircraft is reported so that the other responsibilities can account for those errors in their decision making.

Finally, the top-most row of responsibilities includes Resp-2 and Resp-3. Resp-2 is the responsibility for ensuring there is sufficient capacity to manage the air traffic that needs access to the airspace and can initiate a traffic management program to help manage temporary surges in air traffic if necessary. Resp-3 is the responsibility for receiving proposed trajectory modifications and confirming that an aircraft will have alternate trajectories available if the proposed trajectory modifications are implemented. Resp-3 therefore prevents aircraft from being placed on a trajectory with no options to change it if needed.

Although all five responsibilities in Figure 26 are contained within the "Air Traffic Management (ATM)", this does not necessarily imply that the current ATM system will be used to implement the responsibilities needed to manage UAM aircraft.

4.2.3 Updating the Initial STPA Analysis

Having created the initial conceptual architecture shown in Figure 26, the initial STPA analysis of the NAS can be updated to reflect the design decisions that have been made thus far. To do this STPA update, the losses and hazards remain the same and the UCAs and scenarios are refined to reflect the design details in the conceptual architecture. For example, the UCAs and scenarios identified for the abstract *Coordination* control action can be refined now that *Trajectory Modifications* has been identified as one of the more specific control actions. In addition, new UCAs and scenarios may also be identified. This STPA update therefore provides an opportunity to determine what unsafe behaviors might occur in the conceptual architecture.

When the initial conceptual architecture shown in Figure 26 was analyzed, several instances of missing control elements were identified, and changes were made to the conceptual architecture to address these issues. This section discusses one example of a change that was made based on the updated STPA results and then presents the final conceptual architecture that was created after several rounds of iteration. The full updated STPA can be found in Appendix C.

One design flaw in the initial conceptual architecture that was identified by STPA was the inability to adequately prevent a collision between two aircraft if track and trajectory information is not available for an aircraft before it enters an area of airspace where UAM aircraft are operating. Table 16 shows how the initial STPA was updated to identify this refined scenario and the additional requirement that was derived from it.

Table 16: Example of missing feedback identified by updated STPA analysis

UCA Update	**Original UCA-1.1:** Air Traffic Management does not <u>coordinate</u> the interaction between two aircraft when a collision between them is imminent **Updated UCA-1.1.1:** Resp-1 does not <u>provide Trajectory Modifications</u> when the trajectories of two aircraft are in conflict
Scenario Update	**Original Scenario CS-1.1.2-5:** The Air Traffic Management is not aware of future intended movements of the aircraft and wrongly assumes that the aircraft will continue on their current trajectories. Based on this information, the Air Traffic Management wrongly believes that a collision is not imminent. **Refined Scenario CS-1.1.1-2.1:** Resp-1 is not aware of the planned trajectory of the aircraft because at least one of the two aircraft enters the UAM environment without its tracking and trajectory information having been fully received. This could occur if the aircraft is allowed by Resp-4 to enter the UAM environment before tracking and trajectory information can be fully collected by Resp-5. As a result, Resp-1 either does not know the aircraft is there or has the wrong belief about the trajectory of that aircraft and therefore wrongly believes that no collision is imminent.
Additional Requirement	**Req-88:** ATM system shall ensure that tracking and trajectory information is present for an aircraft before it enters the UAM operating environment *[CS-1.1.1-2.1]*

As can be seen in Table 16, CS-1.1.1-2.1 occurs because the availability of track and trajectory information is not considered by Resp-4 before allowing an aircraft entry into UAM airspace. In other words, there is missing coordination between Resp-4 and Resp-5 to ensure that an aircraft can be adequately tracked, and its trajectory is known before it enters UAM airspace. This missing coordination is highlighted by the orange arrows in the partial control structure shown on the left side of Figure 27.

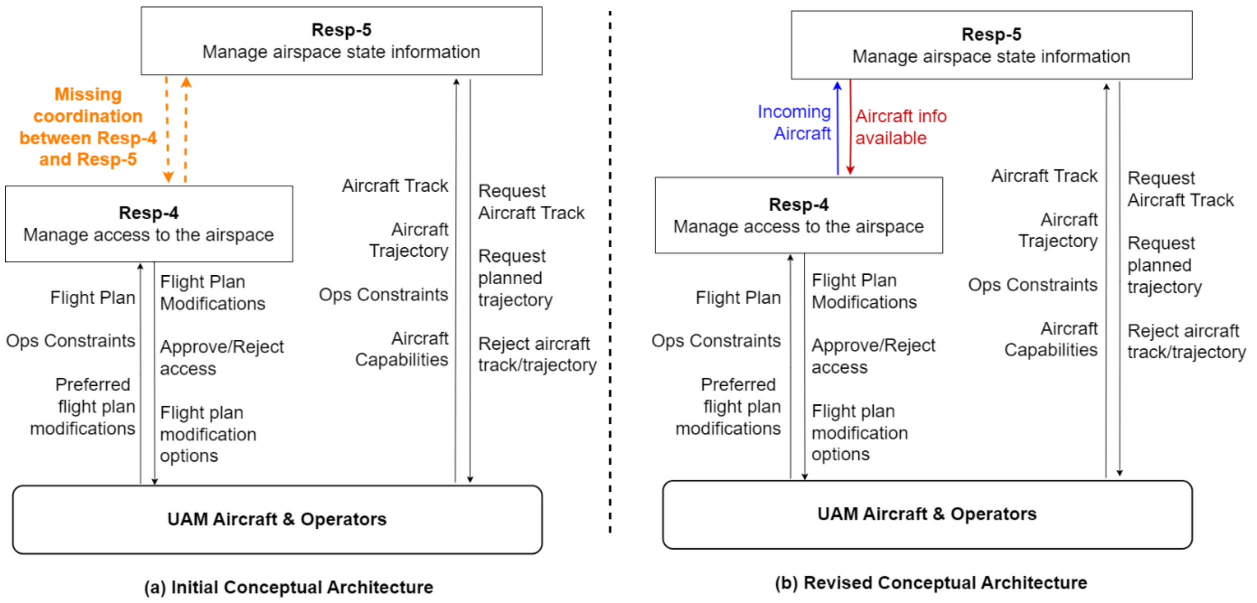
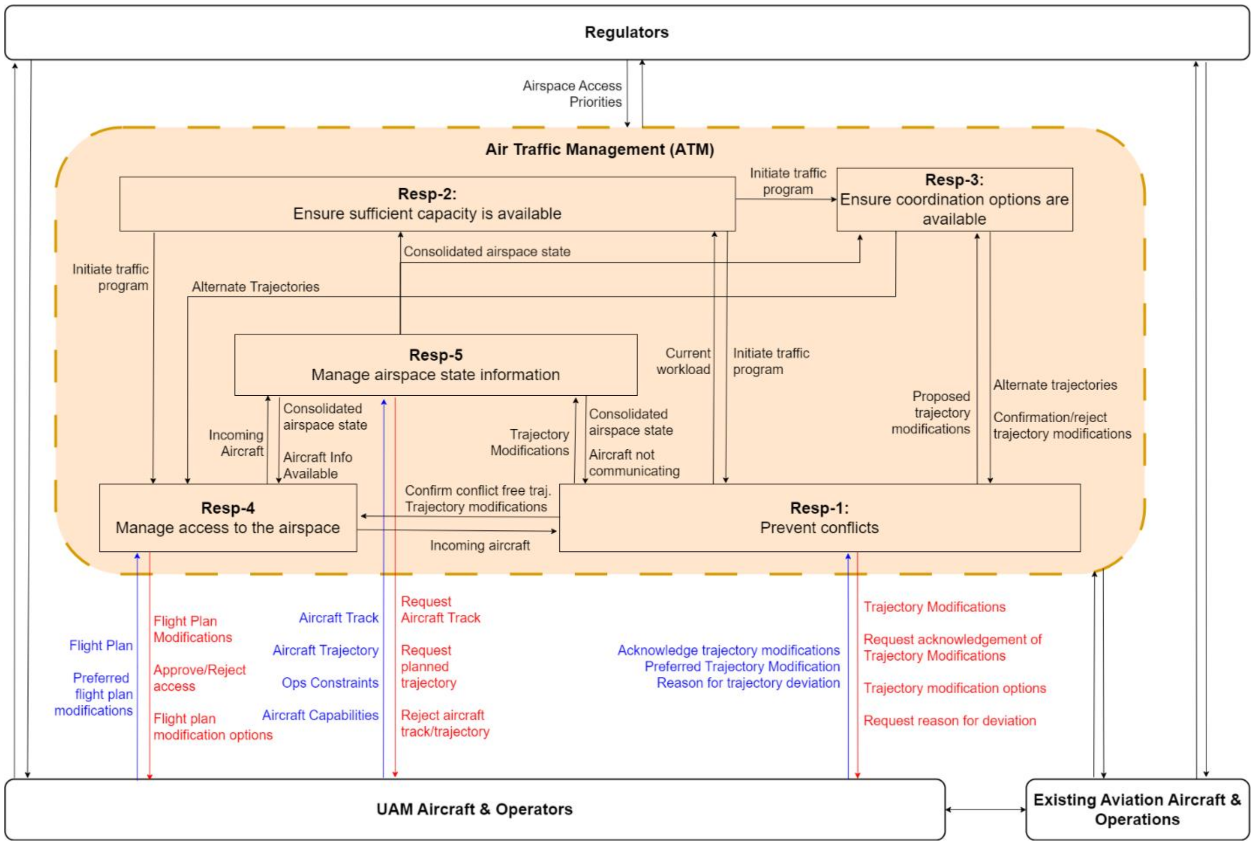

Resp-5
Manage airspace state information
Missing coordination between Resp-4 and Resp-5
Resp-4
Manage access to the airspace
Aircraft Track
Aircraft Trajectory
Ops Constraints
Aircraft Capabilities
Request Aircraft Track
Request planned trajectory
Reject aircraft track/trajectory
Flight Plan
Ops Constraints
Preferred flight plan modifications
Flight Plan Modifications
Approve/Reject access
Flight plan modification options
UAM Aircraft & Operators
(a) Initial Conceptual Architecture
Resp-5
Manage airspace state information
Incoming Aircraft
Aircraft info available
Resp-4
Manage access to the airspace
Aircraft Track
Aircraft Trajectory
Ops Constraints
Aircraft Capabilities
Request Aircraft Track
Request planned trajectory
Reject aircraft track/trajectory
Flight Plan
Ops Constraints
Preferred flight plan modifications
Flight Plan Modifications
Approve/Reject access
Flight plan modification options
UAM Aircraft & Operators
(b) Revised Conceptual Architecture

Figure 27: Zoomed-in view of changes made to Resp-4 and Resp-5 due to Req-88

Thus, to mitigate this refined scenario, Req-88 was added to ensure that track and planned trajectory information are available for all aircraft before an aircraft is allowed access to UAM airspace. The conceptual architecture was then updated to add the necessary control action and feedback between Resp-4 and Resp-5 to meet this new requirement. Thus, as shown on the right side of Figure 27, in the revised conceptual architecture, Resp-4 now provides feedback to Resp-5 about any aircraft inbound to airspace where UAM aircraft are operating and Resp-5 must confirm to Resp-4 that aircraft information (e.g., aircraft track and planned trajectory) is available before Resp-4 allows the aircraft to enter UAM airspace. This modification to the conceptual architecture therefore meets Req-88 and resolves CS-1.1.1-2.1

4.2.4 Revised Collision Avoidance Conceptual Architecture

Figure 28 shows the revised conceptual architecture for collision avoidance that was created. Compared to the initial conceptual architecture shown in Figure 26, this revised conceptual architecture contains the same five responsibilities. However, changes were made to some of the interactions between the responsibilities as well as the control actions and feedback exchanged with the aircraft based on the scenarios identified in the updated STPA.



Regulators
Airspace Access Priorities
Air Traffic Management (ATM)
Resp-2: Ensure sufficient capacity is available
Initiate traffic program
Resp-3: Ensure coordination options are available
Consolidated airspace state
Initiate traffic program
Alternate Trajectories
Resp-5 Manage airspace state information
Current workload
Initiate traffic program
Proposed trajectory modifications
Alternate trajectories
Confirmation/reject trajectory modifications
Incoming Aircraft
Consolidated airspace state
Aircraft Info Available
Trajectory Modifications
Consolidated airspace state
Aircraft not communicating
Confirm conflict free traj.
Trajectory modifications
Incoming aircraft
Resp-4 Manage access to the airspace
Resp-1: Prevent conflicts
Flight Plan
Preferred flight plan modifications
Flight Plan Modifications
Approve/Reject access
Flight plan modification options
Aircraft Track
Aircraft Trajectory
Ops Constraints
Aircraft Capabilities
Request Aircraft Track
Request planned trajectory
Reject aircraft track/trajectory
Acknowledge trajectory modifications
Preferred Trajectory Modification
Reason for trajectory deviation
Trajectory Modifications
Request acknowledgement of Trajectory Modifications
Trajectory modification options
Request reason for deviation
UAM Aircraft & Operators
Existing Aviation Aircraft & Operations

Figure 28: Revised conceptual architecture for collision avoidance

4.3 Exploring and Comparing NAS Architecture Options

Having defined a suitable conceptual architecture for collision avoidance, the structural design process can now be used to identify a NAS system architecture to implement this conceptual architecture.

4.3.1 Identifying Assignment Constraints

First, assignment constraints need to be identified because they inform the architecture options that are created. These assignment constraints are generated using the causal scenarios from the updated STPA analysis of the conceptual architecture that was conducted at the end of the behavioral design process.

The discussion in Section 4.2.3 showed how some of the causal scenarios identified in the STPA analysis of the conceptual architecture could be prevented or mitigated by making changes to the conceptual architecture. However, there were also some scenarios that could not be prevented by changing the conceptual architecture but could be mitigated by making an informed choice about how to assign the various responsibilities. It is from these scenarios that assignment constraints (i.e., preferred assignments) are derived. Table 17 shows four example scenarios, the assignment constraints derived from each scenario and the reasoning that led to each assignment constraint being defined. Additional examples of assignment constraints are shown in Appendix C together with the STPA scenarios.

Table 17: Examples of assignment constraints derived from updated STPA scenarios

UCA-1.1: Resp-1 does not provide Trajectory Modifications when the trajectories of two aircraft are in conflict		
Scenario	**Assignment Constraint**	**Reason for Assignment Constraint**
CS-1.1.1-1.1: The potential conflict is recognized and Resp-1 attempts to resolve the conflict. However, Resp-3 does not confirm that the trajectory modifications still have alternate trajectory options. As a result, Resp-1 is unable to issue trajectory modifications.	Resp-1 = Resp-3	Avoids the need to communicate across controllers in the control structure to receive confirmation that alternate trajectory options are available
CS-1.1.1-1.2: Although feedback about a potential conflict is received, Resp-1 is preoccupied with resolving one set of conflicts and therefore does not issue trajectory modifications to resolve this other conflict.	Resp-1 = Aircraft ∨ (ATM ∧ Aircraft)	Distributing the decision making among the aircraft could help reduce workload and increase decision making capacity
CS-1.1.1-2.3: Resp-1 does not receive feedback about the potential conflict because it does not receive timely feedback on the presence of new ground hazards (e.g., a new construction crane). It therefore does not believe a collision is imminent and does not modify aircraft trajectories.	Resp-1 = Aircraft ∨ (ATM ∧ Aircraft)	Unlike ATM, aircraft have better access to feedback about detected ground hazards and could make faster and more accurate trajectory modification decisions to avoid them
CS-1.29.1-1.3: Although the imminent collision is recognized, the process of generating a resolution repeatedly gets interrupted by new conflicts due to the density of air traffic. As such, before the trajectory modifications can be issued, they need to be recalculated and thus the trajectory of aircraft are not modified until it is too late to avoid a collision	Resp-1 = ATM	Under high density traffic situations, ATM would be better able to anticipate future conflicts and can pre-emptively avoid them instead of only reacting to conflicts as they occur

The scenarios listed in the left column of Table 17 all involve the behavior of Resp-1. For each of these scenarios, it was decided that there was a preferred assignment of Resp-1 that would potentially better mitigate the scenario. For example, scenario CS-1.1.1-1.1 describes inadequate communication between Resp-1 and Resp-3 that would be easiest to prevent if the two responsibilities were assigned to the same controller in the control structure so that they do not need to communicate across controllers. Thus, Resp-1 = Resp-3 is the assignment constraint to indicate this preference for Resp-1 and Resp-3 to be assigned to the same controller.

Similarly, scenario CS-1.1.1-1.2 describes a situation in which a period of high workload while resolving a conflict leads to Resp-1 being unable to resolve a second imminent conflict. This suggests that Resp-1 is vulnerable to disruptions in decision making that could potentially be alleviated if that decision making was either only assigned to the aircraft or shared between ATM and the aircraft so that the aircraft could sometimes help to make those decisions. Thus, the assignment constraint is Resp-1 = Aircraft ∨ (ATM ∧ Aircraft).

From the 55 scenarios identified in the STPA analysis of the conceptual architecture (performed at the end of the behavioral design process), it was determined that 28 of them could potentially be mitigated by a preferred responsibility assignment. Table 18 shows the assignment constraints that were identified and the number of scenarios that could potentially be mitigated by each constraint.

Table 18: Assignment constraints identified from updated STPA scenarios

#	Assignment Constraint	Notation	# of Scenarios
1	Assign Resp-1 and Resp-3 to same controller	Resp-1 = Resp-3	2
2	Assign Resp-1 to ATM	Resp-1 = ATM	5
3	Assign Resp-1 either to UAM aircraft or share it between UAM aircraft and ATM	Resp-1 = Aircraft ∨ (ATM ∧ Aircraft)	21

4.3.2 Creating Architecture Options to Explore

Based on the information in Table 18, assignment constraints 2 and 3 were explored first because they impact the greatest number of scenarios. Based on these two assignment constraints, two candidate architecture options for how to assign Resp-1 were created and the responsibility assignments for each option are shown in Table 19.

Table 19: Responsibility assignments for two architecture options

Resp. ID	Responsibility	**Option A_1** Centralized Collision Avoidance	**Option A_2** Decentralized Collision Avoidance
Resp-1	Identify and resolve conflicts	**ATM**	**Aircraft**
Resp-2	Ensure sufficient capacity is available	ATM	ATM
Resp-3	Ensure coordination options are available	ATM	ATM
Resp-4	Manage access to the airspace	ATM	ATM
Resp-5	Manage airspace state information	ATM	ATM

As highlighted in Table 19, note that the only difference between the two architecture options is the assignment of Resp-1. This was done to ensure that any differences in behavior could be directly attributed to the difference in assignment of Resp-1. For the other four responsibilities, they are assigned to ATM in both architecture options to mirror the architecture that is used in today's air traffic control system.

Architecture option A_1 assigns Resp-1 only to ATM and it represents a centralized collision avoidance architecture where ATM is responsible for identifying and preventing conflicts. This is essentially the same architecture that is used in today's air traffic control system. By contrast, A_2 assigns Resp-1 only to the aircraft and it represents a decentralized collision avoidance architecture where the aircraft are responsible for identifying and preventing conflicts. This architecture is comparable to the Free Flight concept that was proposed by the German Aerospace Center and others in the early 2000s [101]. A_1 and A_2 therefore represent diverse architectures for collision avoidance and comparing them can provide insight into how best to assign the various responsibilities.

To illustrate the differences in these two system architectures, simplified control structures for architecture options A_1 and A_2 are shown in Figure 29 and Figure 30 respectively. Both figures show a zoomed-in version of the control structure in Figure 24 to focus on the differences in the ATM-aircraft and aircraft-aircraft interactions between architecture options. In each figure, Resp-1 is shown in orange and the *Trajectory Modifications* control action (the main control action for Resp-1) is highlighted in red to show how its location in the control structure changes between architecture options.

In addition, because each aircraft has the same interactions with ATM, the control actions and feedback between ATM and the aircraft are only listed for one aircraft in the control structure and it is implied that they apply to the other aircraft as well.

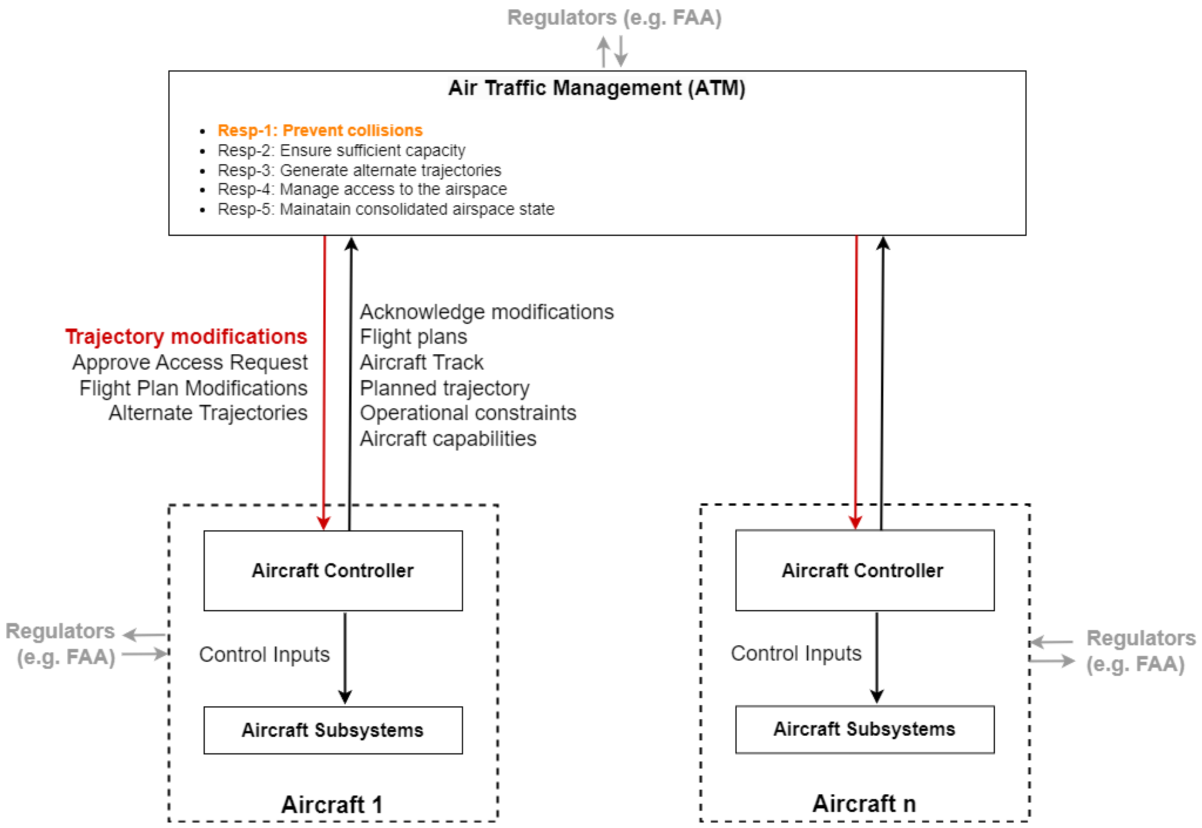


Figure 29: Zoomed-in control structure for architecture option A_1

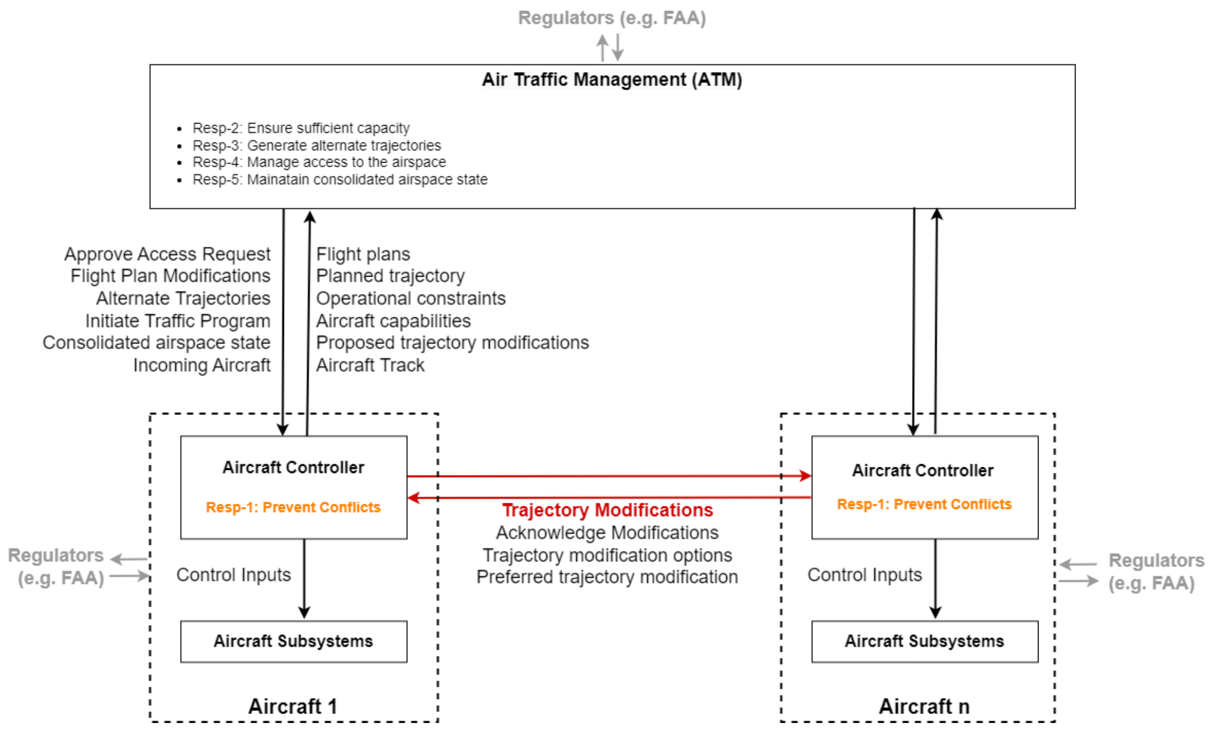

Regulators (e.g. FAA)
Air Traffic Management (ATM)
Resp-2: Ensure sufficient capacity
Resp-3: Generate alternate trajectories
Resp-4: Manage access to the airspace
Resp-5: Mainatain consolidated airspace state
Approve Access Request
Flight Plan Modifications
Alternate Trajectories
Initiate Traffic Program
Consolidated airspace state
Incoming Aircraft
Flight plans
Planned trajectory
Operational constraints
Aircraft capabilities
Proposed trajectory modifications
Aircraft Track
Aircraft Controller
Resp-1: Prevent Conflicts
Trajectory Modifications
Acknowledge Modifications
Trajectory modification options
Preferred trajectory modification
Aircraft Controller
Resp-1: Prevent Conflicts
Regulators (e.g. FAA)
Control Inputs
Control Inputs
Regulators (e.g. FAA)
Aircraft Subsystems
Aircraft Subsystems
Aircraft 1
Aircraft n

Figure 30: Zoomed-in control structure for architecture option A_2

4.3.3 Evaluating and Comparing Architecture Options

Having created these two architecture options, they can now be further analyzed using STPA and compared. For conciseness, this section will focus on the comparison results derived from the STPA analyses of each option. As an example, Table 20 shows four scenarios and the comparison results derived from them for the two architecture options. The full comparison table showing all the STPA scenarios that were used to compare these two architecture options can be found in Appendix D.

Table 20: Architecture comparison table for four example scenarios

#	**Scenario**	**Scenario Occurs?**		**Evaluation Criteria**
		A_1	**A_2**	
1	Although the imminent collision is recognized, <controller(s) performing Resp-1> gets repeatedly interrupted by changing flight conditions and it constantly needs to modify its solution. As a result, a final solution is not selected until it is too late to prevent a collision	**No** *[A1]*	**Yes**	**Responsiveness** of trajectory modifications decisions to prevent loss of separation when flight conditions change rapidly
2	<Controller(s) performing Resp-1> do not receive feedback about the potential conflict because they are unable to receive timely feedback on the presence of new ground hazards (e.g., a new construction crane). As a result, it does not believe a collision is imminent and does not try to modify aircraft trajectories to avoid the collision	**Yes**	**No** *[A2]*	**Timeliness** of ground hazards feedback to prevent loss of separation when resolving a conflict involving terrain or ground obstacles
3	<Controller(s) performing Resp-1> issue trajectory modifications that do not result in collision. However, during transmission to the aircraft, part of the trajectory modification is dropped (e.g., due to a communications error). As a result, the aircraft only receives part of the trajectory modifications and that causes a conflict with another aircraft	**Yes**	**No** *[A3]*	**Vulnerability** of providing trajectory modifications to prevent loss of separation when communication (path) errors occur
4	Although feedback about the potential conflict is received, <controller(s) performing Resp-1> do not issue trajectory modifications because they are preoccupied with resolving one set of conflicts and do not attend to feedback about a subsequent set.	**Yes**	**No** *[A4]*	**Frequency and complexity** of trajectory modifications decisions to prevent loss of separation when resolving a conflict

Table 20 shows that this scenario-based comparison of architecture options can identify evaluation criteria that cover multiple areas of control. The first and fourth criteria involve decision making, the second criterion involves feedback, and the third criterion involves the control path. To illustrate how the evaluation criteria in Table 20 were derived, the difference in behavior of architecture options A_1 and A_2 in the first and second scenarios are illustrated in Figure 31 and Figure 32 respectively.

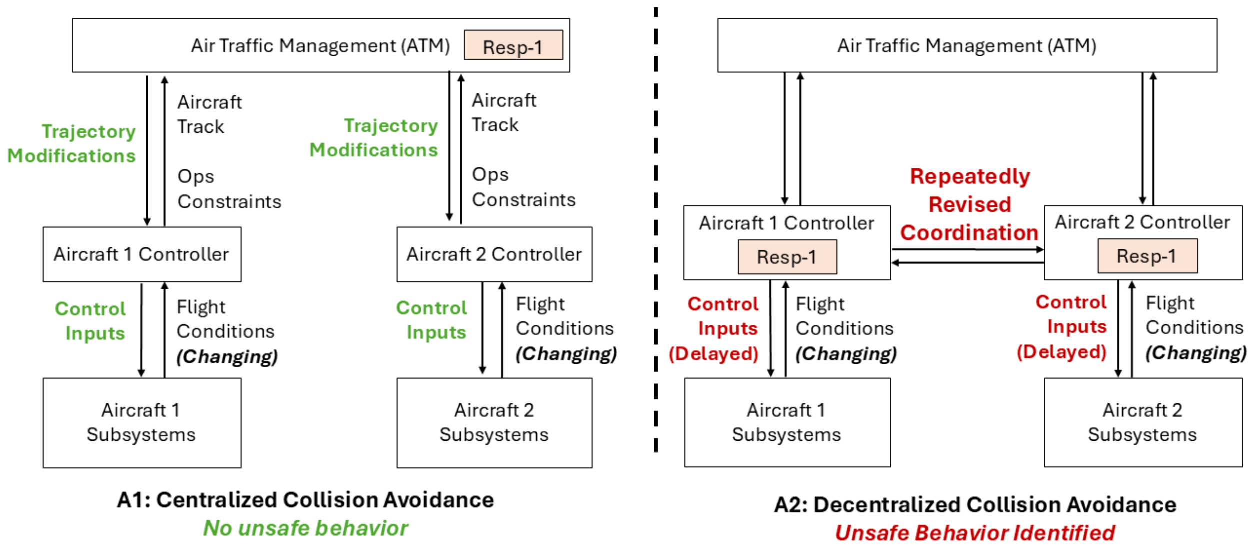
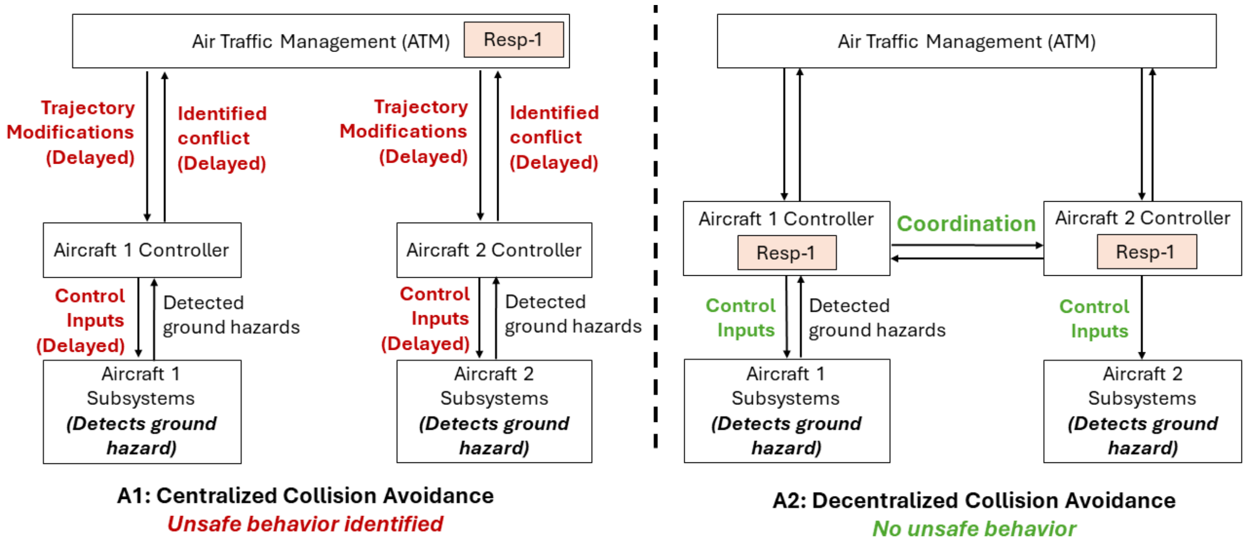


Figure 31: Behavior of A₁ (left) and A₂ (right) in scenario 1 of Table 20



Air Traffic Management (ATM)
Resp-1
Trajectory Modifications (Delayed)
Identified conflict (Delayed)
Trajectory Modifications (Delayed)
Identified conflict (Delayed)
Aircraft 1 Controller
Aircraft 2 Controller
Control Inputs (Delayed)
Detected ground hazards
Control Inputs (Delayed)
Detected ground hazards
Aircraft 1 Subsystems (Detects ground hazard)
Aircraft 2 Subsystems (Detects ground hazard)
A1: Centralized Collision Avoidance
Unsafe behavior identified
Air Traffic Management (ATM)
Aircraft 1 Controller
Resp-1
Coordination
Aircraft 2 Controller
Resp-1
Control Inputs
Detected ground hazards
Control Inputs
Aircraft 1 Subsystems (Detects ground hazard)
Aircraft 2 Subsystems (Detects ground hazard)
A2: Decentralized Collision Avoidance
No unsafe behavior

Figure 32: Behavior of A₁ (left) and A₂ (right) in scenario 2 of Table 20

First, consider scenario 1 (Figure 31). In this scenario, flight conditions (e.g., operational constraints, weather etc.) are changing rapidly and therefore the acceptable conflict resolution options are changing as well. Under these conditions, no unsafe behavior is observed for architecture option A_1 because when Resp-1 is assigned to ATM, ATM is the sole decision maker and can quickly adapt its decision making as the flight conditions change. In addition, ATM has broader situational awareness of the state of the airspace and could anticipate some of these changes. As a result, it can select appropriate trajectory modifications and issue them to the aircraft with minimal delay. By contrast, in option A_2 where Resp-1 is assigned to the aircraft, the aircraft must coordinate with each other to collectively make safe trajectory modification decisions. Therefore, when conditions are changing rapidly, the aircraft will likely need to repeatedly revise their coordination to select appropriate trajectory modifications based on the updated flight conditions. This repeated revision of coordination can therefore delay their selection of trajectory modifications and therefore delay the aircraft in taking appropriate control inputs to resolve the conflict.

Comparing the behavior of architecture options A_1 and A_2 in this first scenario therefore shows that the main behavioral difference between them is the responsiveness with which trajectory modification decisions can be made when flight conditions change rapidly. Specifically, architecture option A_1 enables more responsive decision making than A_2. This therefore leads to the formulation of the evaluation criterion for scenario 1 that is shown in Table 20.

Next, consider scenario 2 (Figure 32). In this scenario, the aircraft detect a ground hazard that was not previously known and recognize that their trajectories conflict with that detected ground hazard. Unsafe behavior is observed for architecture option A_1 because when Resp-1 is assigned to ATM, ATM must first receive feedback about the ground hazard from the aircraft before it can identify the conflict and decide how to resolve it. Thus, because ATM is dependent on the aircraft to provide this feedback about ground hazards, there is a delay before ATM can issue trajectory modifications. Depending on the length of the delay and the distance to the ground hazard, there may not be enough time to resolve the conflict before a collision occurs. By contrast, in option A_2, as soon as the ground hazard is detected, the aircraft can begin coordinating to resolve the conflict and no unsafe behavior occurs due to communications delay.

Thus, comparing the behavior of architecture options A_1 and A_2 in this second scenario shows that the main behavioral difference between them is the timeliness with which ground hazards feedback is received when resolving a conflict involving terrain or ground hazards. Specifically, architecture option A_2 enables more timely feedback about ground hazards than A_1. This leads to the formulation of the evaluation criterion for scenario 2 that is shown in Table 20.

As these decisions are made, it is also important to record any underlying assumptions used to make these decisions. Table 20 also shows that each time it is decided that a scenario does not occur for an architecture option, any assumptions that were used to make that decision are identified and the assumption IDs are indicated in italicized text in the corresponding cell. The assumptions linked in Table 20 are shown in Table 21.

Table 21: Examples of assumptions underlying comparison decisions

ID	**Assumption**
A-1	It is assumed that ATM will not have to coordinate conflicts as frequently because it has broader situational awareness of the future state of the airspace and can better resolve multiple conflicts over longer time horizons in a more coordinated fashion.
A-2	It is assumed that UAM aircraft would have onboard sensing capable of detecting ground hazards with enough range to allow time for the aircraft to respond to avoid a collision with the ground hazard.
A-3	It is assumed that with the aircraft sharing responsibility for preventing conflicts with ATM, a component failure (e.g., on ATM or on one of the aircraft) should not compromise the ability of other aircraft to prevent conflicts.
A-4	It is assumed that even if an initial set of aircraft are preoccupied with resolving a set of conflicts, any new aircraft would identify the conflict and coordinate its own set of trajectory modifications to avoid the other group of aircraft.

As discussed briefly in Chapter 3, it is important to identify these assumptions because the ability of an architecture to prevent a specified scenario is contingent on these assumptions being valid. Thus, if one of these architecture options is chosen for further development, then any downstream design decisions must not violate the assumptions associated with that architecture option. Chapter 6 provides a more detailed discussion of how to ensure that these assumptions remain valid as the design process progresses.

In this design iteration, a total of nineteen evaluation criteria were identified across all aspects of control. These evaluation criteria highlighted key benefits and tradeoffs in three main areas of control: (1) decision making, (2) feedback and control inputs, and (3) control path. The remainder of this section will discuss the benefits and tradeoffs identified in each of these areas of control. The list of all nineteen evaluation criteria can be found in Appendix D.

Decision Making Tradeoffs for Collision Avoidance

The first finding from this comparison is that the two architecture options exhibit important differences in the ability of ATM or the aircraft to make safe and appropriate trajectory modification decisions to resolve conflicts. Table 22 shows the four evaluation criteria that highlight these differences.

Table 22: Comparison results showing decision making tradeoffs for collision avoidance

ID	**Evaluation Criteria**	**Benefit (+) or Tradeoff (-)**	
		A_1	**A_2**
EC-1	**Frequency and complexity** of trajectory modifications decisions when resolving a conflict	⊖	⊕
EC-4	**Ability** to make appropriate trajectory modification decisions to prevent loss of separation when multiple conflicts occur	⊖	⊕
EC-5	**Responsiveness** of trajectory modification decisions to prevent loss of separation when resolving a multi-aircraft conflict in densely populated airspace	⊕	⊖
EC-6	**Responsiveness** of trajectory modifications decisions to prevent loss of separation when the state of the airspace changes rapidly or a conflict involves restrictive operational constraints	⊕	⊖

The first two rows of Table 22 (EC-1 and EC-4) show that when Resp-1 is assigned to the aircraft (as it is in A_2), the ability of the system to make appropriate trajectory modification decisions is improved in two key ways. First, by assigning Resp-1 to the aircraft, the frequency and complexity of the decisions made by each aircraft is lower. This is because allowing the aircraft to perform Resp-1 distributes the decision making for conflict resolution. As a result, multiple groups of aircraft can resolve smaller conflict sets in parallel, making it easier to select appropriate trajectory modifications. This is possible when the traffic density is low because conflicts are more likely to occur further apart in space and therefore can be resolved independently. By contrast, when Resp-1 is only assigned to ATM, ATM is the sole decision maker and must resolve all conflicts of any size that might occur at any time.

The other improvement is that when multiple conflicts occur, they can be better resolved by the aircraft than by ATM. This is because distributing the decision making to the aircraft allows the aircraft to resolve different conflicts in parallel. By contrast, when Resp-1 is assigned to ATM, ATM must resolve all the conflicts by itself.

However, the latter two rows of Table 22 also show these benefits may not be realized by architecture A_2 under some challenging air traffic conditions. EC-5 shows that if the traffic density is high, assigning Resp-1 to ATM (as it is in A_1) allows ATM to make more responsive (i.e., timely) trajectory modification decisions. Similarly, EC-6 shows that if the state of the airspace is changing rapidly or a conflict involves multiple restrictive operational constraints (e.g., an emergency, fuel or battery range limits etc.), assigning Resp-1 to ATM (as it is in A_1) allows ATM to make more responsive trajectory modification decisions. Architecture option A_1 can achieve these benefits because under these more challenging air traffic conditions, it is necessary to coordinate the resolution of these conflicts to avoid causing secondary conflicts. Thus, when Resp-1 is assigned to ATM, ATM's broader situational awareness of the state of the airspace allows it to more easily make multiple simultaneous trajectory modification decisions to resolve many potential conflicts while accounting for all relevant operational constraints. In addition, ATM's broader situational awareness allows it to more easily anticipate future changes to the state of the airspace and pre-empt future conflicts or problems before they can occur. By contrast, if the aircraft are performing Resp-1, their more limited situational awareness of the state of the airspace makes it harder for them to anticipate future conflicts. In addition, their ability to make timely decisions may be impaired by the need to coordinate trajectory modification decisions with other aircraft.

Taken together, the above comparison results show that, when traffic density is low or in less challenging air traffic circumstances, assigning Resp-1 to the aircraft is beneficial because this architecture option lowers the frequency and complexity of trajectory modification decisions and enables better resolution of conflicts when multiple conflicts occur at the same time. However, when air traffic circumstances become more challenging (e.g., high traffic density, strict operational constraints etc.), assigning Resp-1 to ATM enables better coordinated and more timely trajectory modification decisions to be made to resolve any potential conflicts.

Decision Making Tradeoffs for Efficient Management of Airspace

Although this design iteration was primarily focused on safety, several evaluation criteria were also identified that involved efficiency. The comparison of these two architecture options showed that architecture option A_1 enables better decision making for ensuring (1) efficient use of the airspace and (2) that high-priority flights receive the necessary precedence to complete their flights. Table 23 shows the two evaluation criteria (EC-7 and EC-8) that illustrate these differences in behavior.

Table 23: Comparison results showing decision making tradeoffs for efficiency

ID	Evaluation Criteria	Benefit (+) or Tradeoff (-)	
		A_1	A_2
EC-7	**Responsiveness** of trajectory modifications decisions to enable aircraft to complete missions when <u>reducing spacing between aircraft to accommodate additional air traffic</u>	⊕	⊖
EC-8	**Responsiveness** of trajectory modification decisions to inability to complete missions when <u>a high-priority flight needs to be given precedence for mission completion</u>	⊕	⊖

As shown in Table 23, architecture option A_1 exhibits more responsive decision making when making trajectory modifications to reduce the spacing between aircraft and to ensure that a high-priority flight can complete its mission without interference from other, lower-priority air traffic. These two benefits of option A_1 are achieved because, when Resp-1 is assigned to ATM (as it is in A_1), ATM has broader situational awareness of the current and future state of the airspace. As a result, ATM is better equipped than the aircraft to make trajectory modification decisions to either make more efficient use of the airspace to accommodate more aircraft or prioritize a high-priority flight.

In addition, ATM's role in the system is to serve the needs of all airspace users in a fair and consistent manner. Thus, ATM is less likely to act unfairly toward any particular airspace user and will prioritize flights that need that priority in a consistent manner. By contrast, if Resp-1 is assigned to the aircraft, some aircraft may select trajectory modifications that protect their own self-interest (e.g., efficiency of their own trajectory) at the expense of other NAS users.

Feedback and Control Inputs Tradeoffs

Another interesting finding from this comparison was that there is a tradeoff between needing to receive control inputs from other controllers (in the control structure) and the ability to receive timely environmental feedback to inform trajectory modification decisions. Table 24 shows the comparison results for the five relevant evaluation criteria that illustrate this tradeoff.

Table 24: Comparison results showing feedback and control inputs tradeoffs

ID	**Evaluation Criteria**	**Benefit (+) or Tradeoff (-)**	
		A_1	**A_2**
EC-12	**Timeliness** of ground hazards feedback to prevent loss of separation when <u>resolving a conflict</u>	⊖	⊕
EC-13	**Timeliness** of operational constraints feedback to prevent loss of separation when <u>operational constraints are changing frequently</u>	⊖	⊕
EC-14	**Timeliness** of aircraft capabilities, flight conditions and operational constraints feedback to prevent loss of separation when <u>resolving a conflict</u>	⊖	⊕
EC-15	**Use of** "confirmation of trajectory modifications" input to prevent loss of separation when <u>resolving a conflict</u>	⊕	⊖
EC-16	**Use of** "mutual agreement" input to prevent loss of separation when <u>resolving a conflict involving numerous aircraft and/or densely populated airspace</u>	⊕	⊖

The first three rows of Table 24 (EC-12, EC-13, and EC-14) show that when Resp-1 is assigned to the aircraft (as it is in architecture option A_2), one of the benefits is that it is easier for the aircraft to obtain timely feedback on environmental or flight conditions such as aircraft capabilities, flight conditions, operational constraints, and ground hazards. This is especially important if operational constraints or flight conditions are changing frequently. Architecture option A_2 exhibits this benefit because when Resp-1 is assigned to the aircraft, the aircraft have direct access to data about these conditions through on-board sensors (an assumption that is recorded in the comparison results of this architecture option) and direct communication with each other. By contrast, when Resp-1 is assigned to ATM, ATM is dependent on the aircraft or other third-party sources (e.g., weather providing services etc.) to provide feedback about these elements and therefore ATM's ability to receive timely feedback is poorer compared to the aircraft.

However, the last two rows of Table 24 (EC-15 and EC-16) shows that when Resp-1 is assigned to the aircraft (as it is in A_2), one of the tradeoffs is that it becomes necessary to receive two control inputs before trajectory modifications can be selected. The first is confirmation of trajectory modifications. This input is necessary because coordination is needed between Resp-1 and Resp-3 to ensure that a proposed set of trajectory modifications has alternate trajectories available before those trajectory modifications are provided to the aircraft. Recall that in both architecture options, Resp-3 is assigned to ATM. Thus, when Resp-1 is assigned to the aircraft, the aircraft need to wait for ATM to confirm their proposed trajectory modifications before they can execute them, and this could potentially slow down the ability of the aircraft to resolve a conflict. By contrast, when Resp-1 is also assigned to ATM (as it is in A_1), coordination between Resp-1 and Resp-3 occurs within ATM and input from another control element is not required.

The other control input is mutual agreement between aircraft. This input is necessary because when Resp-1 is assigned to the aircraft, the aircraft must work together to coordinate their selection of trajectory modifications. This coordination ensures that the modifications they select do not conflict with each other in addition to not conflicting with other nearby aircraft. By contrast, when Resp-1 is assigned to ATM, this mutual agreement is not needed because ATM is the sole decision maker and need not coordinate its decision with any other control element.

Control Path Benefits of Architecture Option A_2

The last interesting finding from this comparison was that there are some important control path benefits of architecture option A_2 over option A_1 that arise because the decision making associated with Resp-1 is distributed among the aircraft. Table 25 shows the evaluation criteria that illustrate these benefits.

Table 25: Comparison results showing control path tradeoffs

ID	Evaluation Criteria	Benefit (+) or Tradeoff (-)	
		A_1	A_2
EC-17	**Vulnerability** of providing trajectory modifications to prevent loss of separation when a component failure compromises decision making	⊖	⊕
EC-18	**Vulnerability** of providing trajectory modifications to prevent loss of separation when errors with the communications path occurs	⊖	⊕
EC-19	**Responsiveness** of execution of trajectory modifications to prevent loss of separation when trajectory modifications have been issued	⊖	⊕

The first two rows of Table 25 (EC-17 and EC-18) show that when Resp-1 is assigned to the aircraft (as it is in A_2), the control path is less vulnerable to communications errors or component failures that could lead to compromised decision making. Architecture A_2 exhibits this benefit because, when Resp-1 is assigned to the aircraft (as it is in A_2), the inability of one or a group of aircraft to resolve a conflict or accurately transmit trajectory modifications does not necessarily compromise the ability of other aircraft to do so. Thus, even if one aircraft is not communicating, the other aircraft have the capability to maneuver to avoid the non-communicative aircraft, thereby still successfully preventing a collision. By contrast, when Resp-1 is assigned to ATM, if ATM is unable to resolve a conflict or accurately communicate its trajectory modifications to the aircraft, the ability of architecture option A_1 to perform adequate collision avoidance is significantly compromised because no one else is assigned the responsibility to resolve collisions.

In addition, the third row of Table 25 (EC-19) shows that when Resp-1 is assigned to the aircraft, the aircraft are likely to be more responsive in executing the selected trajectory modifications. Architecture A_2 exhibits this additional benefit because, when Resp-1 is assigned to the aircraft, they are the ones selecting trajectory modifications and therefore know that they will soon need to execute those trajectory modifications. By contrast, when Resp-1 is assigned to ATM, ATM could identify and resolve a conflict, but the aircraft may be delayed in executing ATM's selected trajectory modifications if they are preoccupied with other tasks and do not attend to the trajectory modifications right away.

4.4 Evaluation of Comparison Results Against Existing Literature

Having generated a set of benefits and tradeoffs by comparing a centralized and decentralized ATM architecture, the goal of this section is to evaluate whether the framework was able to identify relevant criteria for comparing architecture options. As discussed in Section 2.2, one of the limitations of current architecture development methods is that they are heavily reliant on quantitative metrics, which are challenging to identify during the early stages of development for evaluating emergent properties like safety. Thus, this comparison seeks to determine if this framework overcomes this limitation and can generate relevant qualitative criteria for comparing architecture options.

As discussed at the beginning of this chapter, although this research employs a novel, control-oriented approach to compare architecture options, several studies in the existing literature have already compared centralized and decentralized ATM architectures. In the existing ATM literature, several simulation studies have compared the performance of today's ATC system (a centralized architecture) with several proposed decentralized ATM architectures such as Free Flight and Distributed Air-Ground Traffic Management (DAG-TM). In this section, the benefits and tradeoffs discussed in Section 4.3.3 are compared to those identified in the existing literature to determine (1) if the benefits and tradeoffs found in previous simulation studies are also found using this approach and (2) if this approach is able to identify additional benefits and tradeoffs that are relevant for deciding how best to assign the control responsibilities to achieve emergent properties such as safety and efficiency.

One of the challenges in performing this comparison of results is that the existing studies that this research is being compared to are not of the same type. The framework proposed in this research employs STPA (a qualitative analysis method) to analyze proposed architecture options whereas the existing ATM literature uses simulation studies (a quantitative analysis method). As a result, the centralized and decentralized architectures analyzed in this research are defined at a relatively high level of abstraction whereas the architectures analyzed in the existing ATM literature are defined at a much more detailed level to enable their implementation in a simulation. Consequently, the benefits and tradeoffs identified in this research are qualitative and less detailed while the benefits and tradeoffs identified in the existing literature are typically quantitative and more detailed.

For these reasons, this comparison focuses primarily on the qualitative observations made in the existing literature rather than on the quantitative findings to ensure the same types of findings are being compared. However, to ensure this evaluation does not unfairly discount quantitative results, a qualitative finding is also considered to have been found by the existing literature if it could have reasonably been identified using the quantitative results of a research study. In addition, this comparison focuses only on qualitative observations that are at the same level of abstraction as those identified in this research. Thus, the more detailed benefits or tradeoffs that are identified in some of the existing literature are not considered.

It is also worth noting that this evaluation only includes a limited number of existing research studies and is not meant to be an exhaustive comparison to all studies comparing centralized and decentralized ATM architectures. However, repetition of some of the findings across studies suggests that they are converging toward a common set of benefits and tradeoffs that will be used for this evaluation.

Quantitative Comparison of Benefits and Tradeoffs

Table 26 shows the number of benefits and tradeoffs identified by the existing literature and using the framework developed in this research, grouped by the different aspects of control.

Table 26: Comparing results identified in existing literature and this research

Control Aspect	**Found in Existing Literature**	**Found Using This Framework**
Decision Making	5	8
Process Models	1	3
Feedback and External Inputs	0	5
Control Path	3	3
TOTAL	9	19

Table 26 shows that of the nineteen benefits and tradeoffs identified using the architecture development framework developed in this research (Appendix D), only nine were also identified by the existing literature. At this level of abstraction, all the benefits and tradeoffs that were identified by the existing literature were also found using this framework. Furthermore, except for the control path, this framework identified more benefits and tradeoffs in each of the control aspects than the existing literature. In fact, benefits and tradeoffs related to feedback and external inputs are not identified at all in the existing literature whereas this research identifies five of them. Together, these results suggest that this framework identifies more benefits and tradeoffs with better coverage over the various aspects of control than the methods used in existing research.

This is important because, as discussed at the beginning of Chapter 3, the ability of an architecture to achieve the desired emergent properties is dependent on its ability to adequately control the system's behavior to avoid unsafe behavior. Thus, better coverage over the various aspects of control ensures that all aspects that contribute to achieving adequate control are considered when identifying the benefits and tradeoffs of each architecture option.

While the quantitative differences shown in Table 26 are encouraging, they are not enough by themselves to establish the contributions of this research. This is because, in addition to being able to find more benefits and tradeoffs, it is also important to establish that the benefits and tradeoffs that are found are also relevant for understanding the behavior of an architecture to inform downstream design decisions. For this reason, the remainder of this section qualitatively compares the benefits and tradeoffs identified in this research to those identified in the existing literature. In the remainder of this section, three main qualitative benefits of this architecture development framework are discussed.

Qualitative Comparison 1: Focus on Control-Related Differences

One of the important qualitative differences between the benefits and tradeoffs identified in this research compared to those identified in the existing literature is that those identified in this research are more focused on control-related differences in behavior between the architecture options. To illustrate this, Table 27 shows several benefits and tradeoffs identified in the existing literature (left column) and the equivalent ones identified using this framework (right column).

Table 27: Examples comparing degree of focus on control-related differences

Benefit/Tradeoff Identified in Existing Literature	**Benefit/Tradeoff Identified Using This Framework**
“Decentralized approach had 229% more collisions than the centralized approach” [102, p. 192]	In a centralized collision avoidance architecture, ATM makes more responsive trajectory modifications decisions to prevent loss of separation when the state of the airspace changes rapidly or a conflict involves restrictive operational constraints
“The centralized strategy suppresses the [occurrence of secondary conflicts] over the entire range of traffic densities” [96, p. 325]	
“The centralized conflict resolution produced a total of 3.2% of scenarios with losses of separation […] whereas the decentralized architecture produced a total of 3.4% of scenarios with losses of separation” [103, p. 7]	
For the decentralized free flight concept, “the estimated mean probability of collisions per 20 minutes aircraft flight equals 5.22×10^{-5}, which is equal to a probability of collisions per aircraft flight hour of 1.6×10^{-4}” [104, p. 9], which the authors deem to be a high risk of collision	In a decentralized collision avoidance architecture, the aircraft have more limited situational awareness of airspace state to prevent loss of separation selecting trajectory modifications under challenging air traffic conditions

As shown in Table 27, the benefits and tradeoffs identified in the existing literature are typically focused on the aspects of an ATM system’s behavior that are quantifiable and observable. However, these quantifiable differences in behavior do not necessarily describe the control behavior of an architecture. For example, the left column of Table 27 shows that the differences between centralized and decentralized architectures are typically reported in terms of quantitative metrics such as the number or percentage of conflicts or collisions that were detected or the probability of collision.

By contrast, the benefits and tradeoffs identified using this framework are more focused on the control-related differences in behavior between architecture options. For example, as shown in the first three rows of Table 27, instead of just observing that the decentralized architecture had more collisions or more secondary collisions than the centralized architecture, this framework identifies that it is ATM’s ability to make more responsive (i.e., more timely) decisions in a centralized architecture that enables it to more adequately resolve any potential conflicts. Similarly, in the last row of Table 27, instead of just observing that a decentralized architecture has a higher probability of collision, this framework identifies that it is because the aircraft have more limited situational awareness of the airspace state that they are more likely to inadequately resolve a conflict.

Qualitative Comparison 2: Determining the Source of Observed Behavioral Differences

Another important qualitative difference between the benefits and tradeoffs identified in this research compared to those identified in the existing literature is that it is easier to determine what aspect of the ATM architecture contributed to the identified benefit or tradeoff. To illustrate this difference, consider the following two examples from the existing literature.

First, in [104], the authors offer the following explanation for their finding that the decentralized free flight concept has a high risk of collision:

There appeared to be five different collision events. […] Four of the five collisions were due to a growing number of multiple conflicts that could not be solved in time under the operational concept adopted. [104, p. 9]

In this first example, although the authors of [104] describe that the collisions occurred due to the growing number of multiple conflicts occurring, they don't explain why their decentralized free flight concept could not resolve those conflicts.

The second example is in [102], where the authors offer the following explanation for why the decentralized approach had more collisions than the centralized approach:

This can be explained by the increased number of messages required by this approach, which [is] associated with the communications overhead, [resulting] in a larger time to reach an agreement [102, p. 192]

In this example, the authors of [102] include in their explanation the element of their decentralized architecture that contributed to the observed increase in collisions (i.e., the communications overhead). However, they do not offer further explanation of why the communications overhead occurs.

These two examples illustrate that in the existing literature, it can be difficult to determine what aspects of the ATM architecture contributed to those benefits and tradeoffs. By contrast, the framework developed in this research makes it easier to determine why an identified benefit or tradeoff occurs because each one is derived from a specific STPA causal scenario that includes details about what control structure elements are involved. Thus, each benefit and tradeoff can be easily traced to specific element(s) in the system architecture. For example, Table 28 shows one of the benefits of a centralized architecture that was identified in this research and the causal scenario that it was derived from.

Table 28: Demonstration of how a causal scenario explains an identified benefit

Benefit	A centralized collision avoidance architecture exhibits more responsive trajectory modifications decisions to prevent loss of separation when the state of the airspace changes rapidly or a conflict involves restrictive operational constraints
Associated STPA Causal Scenario	Although the imminent collision is recognized, <controller(s) performing Resp-1> gets repeatedly interrupted by new conflicts due to the density of air traffic. As such, before the trajectory modifications can be issued, they need to be recalculated and thus the trajectories of aircraft are not modified until it is too late to enact the new trajectories to avoid a collision
Comparison Result for This Scenario	**DOES NOT** occur in architecture option A_1 (centralized architecture) **DOES** occur in architecture option A_2 (decentralized architecture)

Table 28 illustrates how the causal scenario associated with a benefit or tradeoff makes it easier to determine how one of the architectures might exhibit better or worse behavior. One of the benefits of a centralized collision avoidance architecture identified by this framework is that it exhibits responsive trajectory modification decisions when the state of the airspace changes rapidly or a conflict involves restrictive operational constraints. This benefit is observed because when ATM resolves conflicts centrally, it can pre-emptively resolve multiple conflicts simultaneously without needing to interrupt its decision making for each new conflict as they occur. Therefore, no unsafe behavior related to this causal scenario occurs in architecture option A_1. By contrast, when the aircraft resolve their own conflicts, any new conflicts or changes to the conflicts they are resolving will interrupt their decision making and require them to re-evaluate their trajectory modification decisions. Thus, unsafe behavior related to this causal scenario does occur in architecture option A_2.

Qualitative Comparison 3: Consideration of Different Air Traffic Contexts

Finally, the last qualitative difference between the benefits and tradeoffs identified in this research compared to those identified in the existing literature is that the ones identified in this research were derived from a broader consideration of different air traffic contexts. This difference was observed because many of the simulation studies in the existing literature that were considered in this comparison (e.g., [96, 102, 105, 106]) only considered nominal air traffic conditions involving varying levels of air traffic density. Only [97, 101, 103] considered off-nominal conditions such as input/output errors, component failures, delays or flight crews not flying the aircraft according to the trajectories needed to prevent a collision.

By contrast, the scenarios generated in the various STPA analyses performed in this design iteration included the following air traffic contexts:

- High or low traffic density
- Nominal, degraded, and emergency conditions
- Interactions between different types of air traffic/aircraft (e.g., emergency response flights, high-priority flights, etc.)
- Inclement weather conditions
- Airspace that includes restrictions (e.g., temporary flight restrictions (TFRs))

This framework enables benefits and tradeoffs to be derived from a broader consideration of air traffic contexts because architecture options are compared based on the qualitative causal scenarios generated by STPA. Thus, different air traffic contexts (e.g., traffic density, weather conditions etc.) and different combinations of those contexts can be easily included in the UCAs and causal scenarios when performing STPA. This is more challenging to do in a simulation study because each new context or combination of contexts requires an additional execution of the simulation or additional development effort to incorporate the new context or combination of contexts into the simulation.

4.5 Designing the Preferred Collision Avoidance Architecture

Finally, to conclude this design iteration, a preferred collision avoidance architecture for managing UAM air traffic needs to be designed based on the insights gained from comparing architecture options A_1 and A_2.

As discussed in Section 4.3.3, architecture option A_1 exhibits benefits that are desirable if the airspace is anticipated to routinely have high air traffic density or challenging air traffic conditions. This is because in architecture option A_1, assigning Resp-1 to ATM centralizes the responsibility for resolving conflicts with ATM. This allows ATM to make better coordinated, more timely, and more responsive conflict resolution decisions because it has the necessary situational awareness of the overall state of the airspace to resolve multiple conflicts simultaneously. ATM can also more easily anticipate future changes to the state of the airspace and act accordingly to pre-empt future conflicts or problems before they occur.

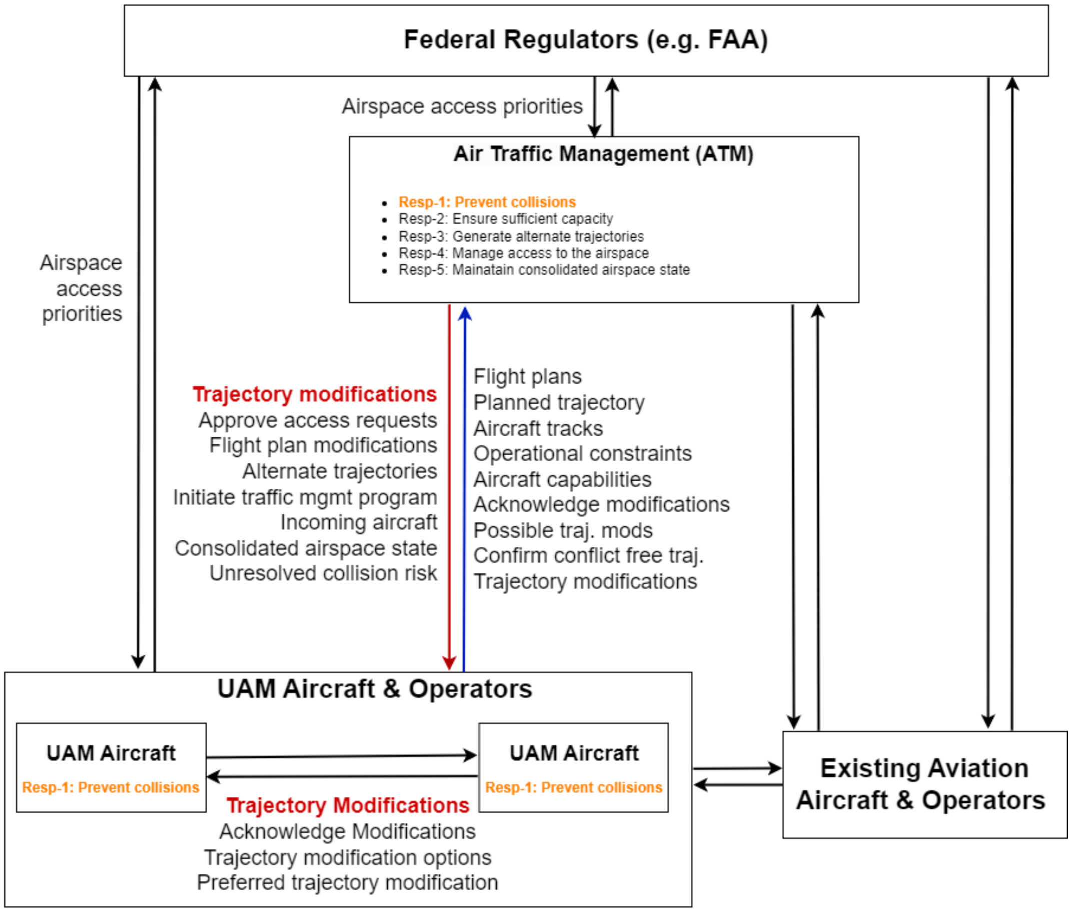
However, the tradeoffs associated with architecture option A_1 are that ATM is required to make complex and frequent conflict resolution decisions because it is solely responsible for preventing all potential conflicts. In addition, the control path for issuing trajectory modifications is vulnerable to disruptions because ATM must make all conflict resolution decisions and be able to transmit appropriate trajectory modifications to the aircraft. Thus, any disruption in ATM's ability to make timely decisions or transmit trajectory modifications to the aircraft would significantly compromise its ability to adequately resolve conflicts.

On the other hand, Section 4.3.3 also identified that architecture option A_2 exhibits benefits that are desirable if the airspace is anticipated to routinely have only low air traffic density. This is because when the air traffic density is low, less coordination is required to resolve different sets of conflicts. Thus, because architecture option A_2 distributes the responsibility for conflict resolution to the aircraft, they can make less complex and less frequent conflict resolution decisions by resolving different sets of conflicts in parallel instead of resolving them all simultaneously. In addition, the control path is less vulnerable because the inability of some aircraft to perform adequate collision avoidance does not compromise the ability of other aircraft to do so. Furthermore, the aircraft can more easily receive timely feedback about aircraft capabilities, flight conditions, and operational constraints. They can therefore make quicker decisions to modify their trajectories in response to changing environmental conditions.

However, the main tradeoff associated with A_2 is the need for the aircraft to coordinate among themselves to select appropriate trajectory modification decisions. Because the aircraft must coordinate among themselves to select appropriate trajectory modifications, how quickly the aircraft can make trajectory modification decisions depends significantly on (1) the number of aircraft involved in the conflict, (2) the density of the surrounding airspace, and (3) the number of operational constraints of the aircraft involved in the conflict that must be considered. If any of these factors are high (e.g., conflict involving many aircraft, dense airspace, numerous tight operational constraints etc.), the aircraft may be significantly slower than ATM at making trajectory modification decisions because they need to coordinate those decisions.

Unfortunately, the traffic conditions in UAM are unlikely to be predictably high density or low density at any given time. This is because the on-demand nature of UAM flights means that flights may occur with limited advance notice. As a result, there may be times when air traffic is unexpectedly light, and the behavior of the decentralized architecture would be preferable. However, there may be other times when a sudden unexpected surge in air traffic causes a period of very high traffic density, and the behavior of the centralized architecture would be preferable.

For this reason, this research proposes a hybrid of the two architecture options. Instead of just selecting either A_1 or A_2, this research proposes a more flexible shared collision avoidance architecture (designated as architecture option A_3) where Resp-1 is assigned to either ATM or the aircraft. By sharing the collision avoidance responsibility between ATM and the aircraft, either ATM or the aircraft could decide to resolve a conflict depending on who is better equipped to make the necessary decisions in each situation. The full control structure for architecture option A_3 is shown in Figure 33.



Federal Regulators (e.g. FAA)
Airspace access priorities
Air Traffic Management (ATM)
Resp-1: Prevent collisions
Resp-2: Ensure sufficient capacity
Resp-3: Generate alternate trajectories
Resp-4: Manage access to the airspace
Resp-5: Mainatain consolidated airspace state
Airspace access priorities
Trajectory modifications
Approve access requests
Flight plan modifications
Alternate trajectories
Initiate traffic mgmt program
Incoming aircraft
Consolidated airspace state
Unresolved collision risk
Flight plans
Planned trajectory
Aircraft tracks
Operational constraints
Aircraft capabilities
Acknowledge modifications
Possible traj. mods
Confirm conflict free traj.
Trajectory modifications
UAM Aircraft & Operators
UAM Aircraft
Resp-1: Prevent collisions
UAM Aircraft
Resp-1: Prevent collisions
Trajectory Modifications
Acknowledge Modifications
Trajectory modification options
Preferred trajectory modification
Existing Aviation Aircraft & Operators

Figure 33: Control structure for architecture option A_3

Because the traffic circumstances in UAM are likely to be dynamic and unpredictable, the benefit of architecture option A_3 (where ATM and the aircraft share responsibility for preventing conflicts) is that it could dynamically allocate that responsibility depending on the prevailing air traffic circumstances. For example, when traffic density is high or an emergency arises, ATM could resolve conflicts instead of the aircraft. This would allow A_3 to be capable of making responsive and better coordinated trajectory modification decisions in high density traffic situations like in A_1. However, if only a few aircraft are involved in an isolated conflict, the aircraft could resolve that conflict instead of ATM. This would allow A_3 to offer the same benefits as A_2. These benefits include a reduction in decision making complexity and frequency when selecting trajectory modifications, the ability to receive timely feedback about environmental conditions and a less vulnerable control path for providing trajectory modifications.

It is worth noting that these benefits of A_3 are contingent on two key assumptions. First, it is assumed that A_3 can be feasibly implemented during detailed system design and that none of the required system elements are impossible to design or implement. Second, it is also assumed that there are no significant tradeoffs of A_3 (e.g., mode confusion) that might negate the expected benefits. These two assumptions are important because if either of these assumptions are not true, the expected benefits of this shared collision avoidance architecture may not be realized, or it will require far more development effort to realize them than anticipated.

These two design assumptions therefore represent two open design issues that need to be explored to confirm that architecture option A_3 (the shared collision avoidance architecture) is the preferred ATM architecture for UAM. For this reason, a second design iteration was conducted in this research to refine the shared collision avoidance architecture to determine the feasibility of implementing it and better understand any potential tradeoffs. This work will be presented next in Chapter 5.

4.6 Summary

This chapter described the results of the first of two design iterations performed in this research. In this first design iteration, the goal was to apply the safety-driven architecture development framework described in Chapter 3 to develop an initial collision avoidance architecture that will be able to safely manage UAM air traffic. To do this, the NAS was first analyzed at a high level of abstraction to identify a set of system-level collision avoidance requirements. Those requirements were then used to create a conceptual architecture that defined the control behavior that would be needed to ensure safe operation with respect to collision avoidance.

Two architecture options were then created and compared to determine the preferred way to implement the conceptual architecture in the ATM system architecture. Architecture option A_1 was a centralized architecture option where ATM was responsible for collision avoidance, and architecture option A_2 was a decentralized architecture option where the aircraft were responsible for collision avoidance.

The comparison of these two architecture options showed that the preferred architecture option depended on the air traffic circumstances that would be encountered. If traffic density was expected to be high, architecture option A_1 would ensure better coordinated and timely trajectory modification decisions. By contrast, if traffic density was expected to be low, architecture option A_2 would enable reduced decision-making complexity and a less vulnerable control path for providing trajectory modifications.

These benefits and tradeoffs were then compared to the benefits and tradeoffs identified by comparisons of centralized and decentralized ATM architectures in the existing literature. This comparison showed that this framework identifies more benefits and tradeoffs that covered more areas of control than the existing literature. In addition, a qualitative comparison of the benefits and tradeoffs identified by this research to those identified in the existing literature showed that this framework offers three additional benefits. First, this framework generates benefits and tradeoffs that are more focused on control-related differences which enhances safety understanding. Second, this framework makes it easier to determine what aspects of the architecture give rise to the observed benefits and tradeoffs. Third, this framework allows benefits and tradeoffs to be derived from a broader consideration of different air traffic contexts.

These results therefore provide support for hypothesis 1 of this dissertation and demonstrate that the framework developed in this research can identify relevant criteria for comparing architecture options and evaluating the architectures' ability to achieve emergent properties.

Hypothesis 1: A systems-theoretic approach can identify relevant criteria for comparing architecture options and evaluating their ability to achieve emergent properties

Finally, to conclude this first design iteration, the preferred collision avoidance architecture for UAM was designed. Because UAM flights are expected to occur on-demand instead of being scheduled in advance, it is unlikely that traffic density will always be predictably high or low. Thus, based on the insights gained from comparing architecture options A_1 and A_2, architecture option A_3 was proposed that assigns the responsibility for collision avoidance to both ATM and the aircraft. This architecture option would enable the ATM system to adapt its collision avoidance behavior and vary the extent to which identified conflicts are resolved by ATM or the aircraft to suit the prevailing air traffic circumstances.

However, the selection of architecture option A_3 assumes that it can be feasibly implemented and that there are no significant tradeoffs that would compromise the safety of A_3 and negate its expected benefits. It is therefore important to validate these assumptions to confirm if A_3 is indeed the preferred ATM architecture for UAM. Thus, a second design iteration is needed to refine architecture option A_3 to better understand the feasibility of its implementation and its tradeoffs. The next chapter presents the results from this second design iteration.

Chapter 5 Design Iteration 2: Refining the Collision Avoidance Architecture

In design iteration 1, the goal was to select the collision avoidance architecture for managing UAM air traffic in the NAS. By applying the architecture development framework developed in Chapter 3, a shared collision avoidance architecture was selected where the responsibility for resolving conflicts (Resp-1) is shared between ATM and the aircraft. However, the selection of this architecture as the preferred option was contingent on two key assumptions: that it could be feasibly implemented and that the tradeoffs would not outweigh the expected benefits.

So, to investigate the validity of these assumptions, a second design iteration was conducted to refine the shared collision avoidance architecture that was created in design iteration 1. The goal of design iteration 2 can therefore be stated as follows:

Iteration 2 Goal: Define how ATM and the aircraft will share responsibility for collision avoidance and work together to adequately resolve conflicts

This chapter presents the results of this second design iteration. First, a past accident is introduced to illustrate how unsafe behavior can occur in an existing ATM architecture with shared responsibility for collision avoidance. Then, the shared collision avoidance architecture developed in design iteration 1 is analyzed using STPA to identify how unsafe behavior could occur. The results from this STPA analysis are then used to develop a refined conceptual architecture for shared collision avoidance. Next, potential architecture options for implementing this refined conceptual architecture are created and compared to identify the potential benefits and tradeoffs. Finally, the results from both design iterations are reviewed to evaluate the ability of the framework to help systems engineers incrementally refine a system architecture and make more informed design decisions.

5.1 Illustration of Unsafe Behavior: The Uberlingen Accident

The shared collision avoidance architecture developed in design iteration 1 represents a more collaborative approach than exists today for sharing the responsibility for collision avoidance between ATM and the aircraft. However, the concept of sharing collision avoidance responsibility between a ground-based controller and an airborne controller is not entirely new. The use of the Traffic Collision Avoidance System (TCAS) onboard commercial aircraft to augment the collision avoidance capabilities of ground-based Air Traffic Control (ATC) is a simpler version of a shared collision avoidance architecture that is already operating in the NAS. Although this system has helped to prevent numerous mid-air collisions, some have still occurred. It is therefore informative to study what could go wrong in this simpler version of shared collision avoidance involving ATC and TCAS to inform the hazard analysis of what could go wrong in the more collaborative shared collision avoidance architecture that this research is proposing to use for UAM.

Background: Brief Overview of TCAS

TCAS is a collision avoidance system installed on commercial aircraft that is designed to help identify and prevent mid-air collisions between aircraft [107]. It does this by monitoring the airspace around an aircraft and warns pilots if a nearby aircraft presents a collision risk. Although

ATC's primary role is to maintain separation between aircraft, TCAS was designed to serve as a backup collision avoidance system in cases when a conflict is not identified and resolved by ATC.

When a potential conflict is identified by TCAS, there are two types of alerts it can issue [107]. The first is called a traffic advisory (TA) that simply warns the flight crew that a nearby aircraft may be a collision risk, but no action is required to be taken. The other type of alert is a resolution advisory (RA) that includes a recommended maneuver (to either climb or descend) to prevent a collision. FAA regulations and the operations manuals of many airlines require that if flight crews receive an RA from TCAS, they are to execute the recommended maneuver even if that RA is contradictory to an instruction provided by ATC. Figure 34 shows a simplified control structure that illustrates how TCAS works within the ATC system.

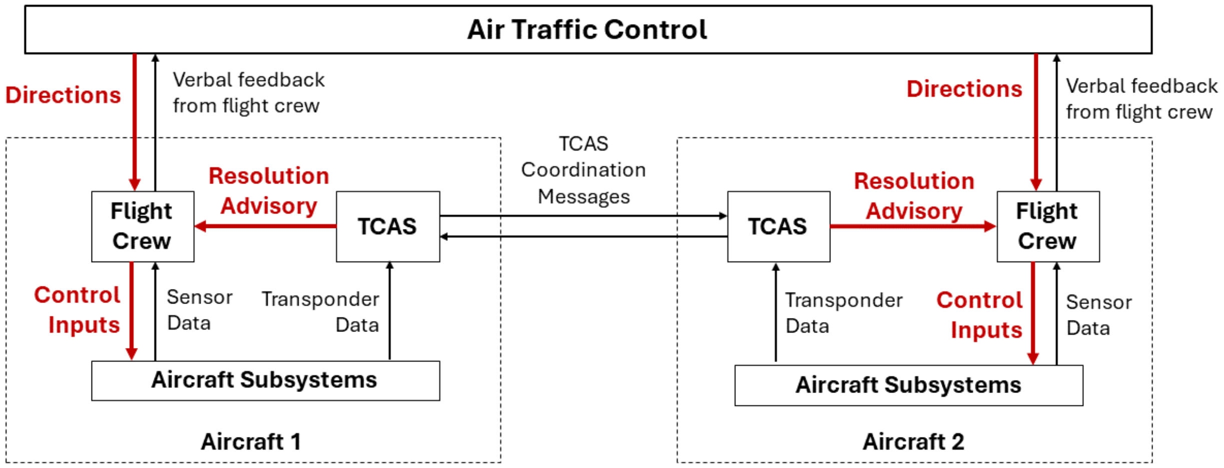


Figure 34: Simplified control structure of today's ATC including TCAS

Although ATC and TCAS are both responsible for identifying and resolving collisions, their responsibilities are not completely overlapping because TCAS only attempts to resolve conflicts that might occur within a limited period of time. By contrast, ATC will attempt to resolve any conflict that it identifies. Thus, if ATC identifies a conflict well in advance of a collision occurring, ATC will provide directions (e.g., climbs, descents, turns) to the flight crew and the flight crew will provide control inputs to the aircraft to carry out ATC's directions. However, if ATC does not identify a conflict soon enough, TCAS might identify that conflict and issue an RA to resolve the conflict. If an RA is issued, the flight crew similarly will need to provide control inputs to the aircraft to comply with the RA.

Figure 34 illustrates two noteworthy features of how TCAS is integrated into the ATC system. First, there is no direct feedback or communication of any kind between TCAS and ATC. TCAS therefore operates entirely independently of ATC and the only way ATC receives feedback about a TCAS RA is when the flight crew provides that feedback verbally via radio communications.

The other noteworthy feature is that this architecture makes it possible for flight crews to receive conflicting instructions about how to maneuver the aircraft and they must decide which instructions to execute. For example, if a conflict is identified at the same time by both ATC and TCAS, ATC and TCAS could choose different ways to maneuver the aircraft to resolve the conflict. This would result in the flight crews receiving conflicting instructions. To avoid confusion, flight crews are expected (by procedure) to always comply with the TCAS RA, even if that means ignoring ATC instructions. This policy ensures a consistent set of instructions are executed.

The Uberlingen Accident

On 1 July 2002, Bashkirian Airlines flight 2937 and DHL flight 611 collided with each other over Uberlingen, Germany while under the control of Zurich Air Traffic Control [108]. Just prior to the accident, both aircraft were cleared to cruise at 36000 feet but the Zurich air traffic controller handling those flights did not realize that he had cleared two aircraft on conflicting flight paths to cruise at the same altitude.

Less than a minute before the crash, the Zurich air traffic controller recognized that the two aircraft were in conflict and chose to resolve it by keeping flight 611 at 36000 feet and instructing flight 2937 to descend. This decision alone was adequate and would have resolved the conflict. However, very shortly after flight 2937 begins its descent, the TCAS on both aircraft identified the same conflict. Unfortunately, because TCAS operates independently of ATC, TCAS chose a different way to resolve the conflict and instructed flight 611 to descend while instructing flight 2937 to climb. As a result, although the flight crew of flight 611 only received instructions from TCAS to descend, the flight crew of flight 2937 received conflicting instructions: an instruction from Zurich ATC to descend and an instruction from TCAS to climb [108]. This scenario is illustrated in the simplified control structure shown in Figure 35.

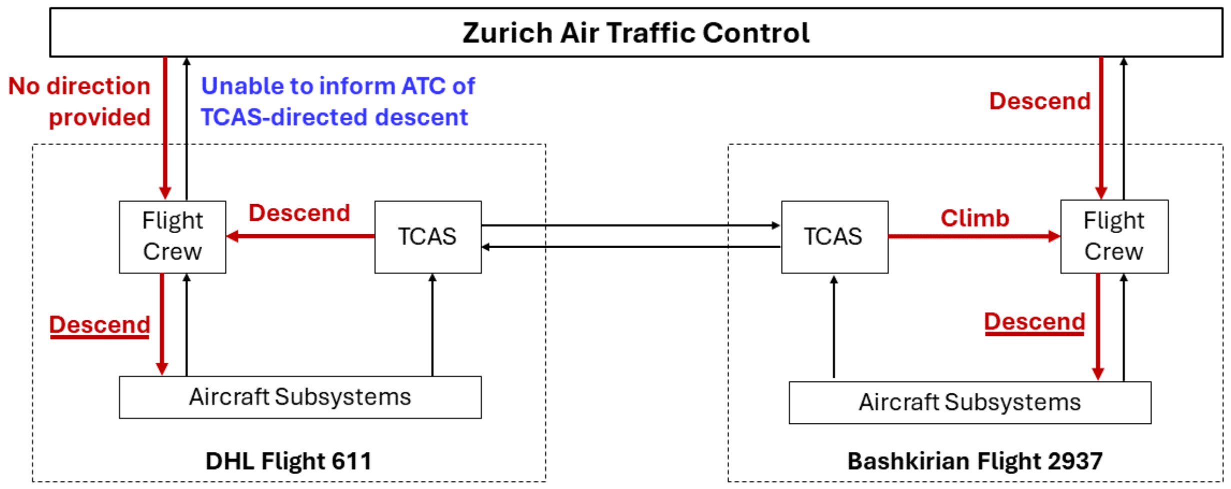


Figure 35: Control structure showing conflicting instructions in the Uberlingen accident

Faced with this scenario, the flight crew of flight 611 followed established procedures and complied with the RA from TCAS instructing them to descend. Unfortunately, because the flight crew of flight 2937 had been trained differently than the flight 611 flight crew, they did not follow the same procedure of always complying with the TCAS RA. Instead, they chose to ignore the TCAS RA and follow the instruction from Zurich ATC to descend. As a result, both aircraft descended toward each other, resulting in a mid-air collision [108].

As has been published in several reports following the accident [108, 109], there were numerous factors that contributed to the accident. However, focusing on the role of the ATC architecture in this accident, two factors are relevant. First, because TCAS and ATC operate independently of each other, TCAS provides no feedback to ATC if an RA is generated, and they do not coordinate to decide if TCAS or ATC is resolving the conflict. Instead, both TCAS and ATC independently try to resolve the conflict and it is up to the flight crew to resolve any conflicting instructions that are provided to them.

Second, this architecture for integrating ATC and TCAS depends on the flight crews consistently following a fixed procedure and always complying with the TCAS RA. Thus, when the flight crews do not follow this procedure consistently (in this case due to differences in training), this Uberlingen accident becomes possible.

Implications for designing the shared collision avoidance architecture

The Uberlingen accident illustrates how even in a relatively simple shared collision avoidance architecture, independent decision making and conflicting instructions can lead to unsafe conflict resolution decisions and inadequate collision avoidance. This suggests that for the more collaborative shared collision avoidance architecture selected in design iteration 1, it will be even more important to design it to avoid unsafe behavior because it contains more opportunities for inadequate collision avoidance than today's ATC system. This is because, compared to TCAS, the shared collision avoidance architecture requires a more advanced airborne collision avoidance system that is given larger scope and expanded authority to resolve conflicts. Unlike TCAS, the shared collision avoidance architecture selected in design iteration 1 allows the aircraft to resolve any conflict in trajectories instead of just conflicts within a limited radius of the aircraft's current position. In addition, aircraft can make any change to their trajectory instead of just climbs or descents.

For these reasons, the purpose of the STPA analysis that will be presented in the next section is to analyze the shared collision avoidance architecture to obtain some initial information about how unsafe behavior like what happened in the Uberlingen accident could lead to inadequate collision avoidance. Those analysis results can then be used to refine the conceptual architecture to ensure that this shared collision avoidance architecture is designed to prevent accidents like the Uberlingen accident from occurring.

5.2 STPA Analysis of Shared Collision Avoidance Architecture

To refine the shared collision avoidance architecture, the STPA analysis that was performed in design iteration 1 is updated again to consider all the ways in which ATM and the aircraft might be unable to adequately resolve conflicts. As with previous STPA analyses in this research, the losses and hazards are the same as those identified in Table 7 and Table 8 respectively. However, a zoomed-in version of the control structure created in iteration 1 (shown in Figure 33) is used because the goal of this analysis is specifically to refine how ATM and the aircraft will need to work together to adequately resolve collisions. Figure 36 highlights the area of the control structure from iteration 1 that will be zoomed-in on and Figure 37 shows the zoomed-in version of the control structure that will be used in this design iteration.

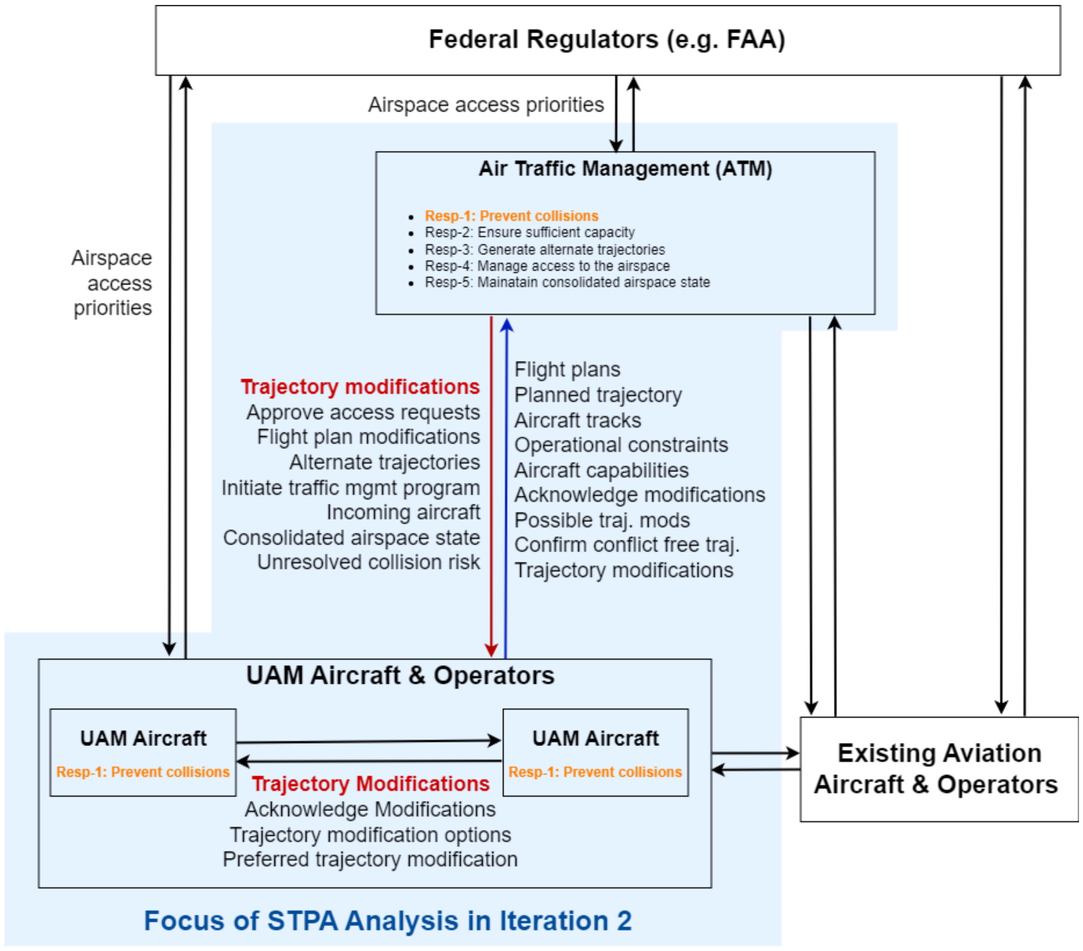

Federal Regulators (e.g. FAA)
Airspace access priorities
Airspace access priorities
Air Traffic Management (ATM)
Resp-1: Prevent collisions
Resp-2: Ensure sufficient capacity
Resp-3: Generate alternate trajectories
Resp-4: Manage access to the airspace
Resp-5: Mainatain consolidated airspace state
Trajectory modifications
Approve access requests
Flight plan modifications
Alternate trajectories
Initiate traffic mgmt program
Incoming aircraft
Consolidated airspace state
Unresolved collision risk
Flight plans
Planned trajectory
Aircraft tracks
Operational constraints
Aircraft capabilities
Acknowledge modifications
Possible traj. mods
Confirm conflict free traj.
Trajectory modifications
UAM Aircraft & Operators
UAM Aircraft
Resp-1: Prevent collisions
UAM Aircraft
Resp-1: Prevent collisions
Trajectory Modifications
Acknowledge Modifications
Trajectory modification options
Preferred trajectory modification
Existing Aviation Aircraft & Operators
Focus of STPA Analysis in Iteration 2

Figure 36: Illustration of area in higher-level control structure that will be zoomed in on

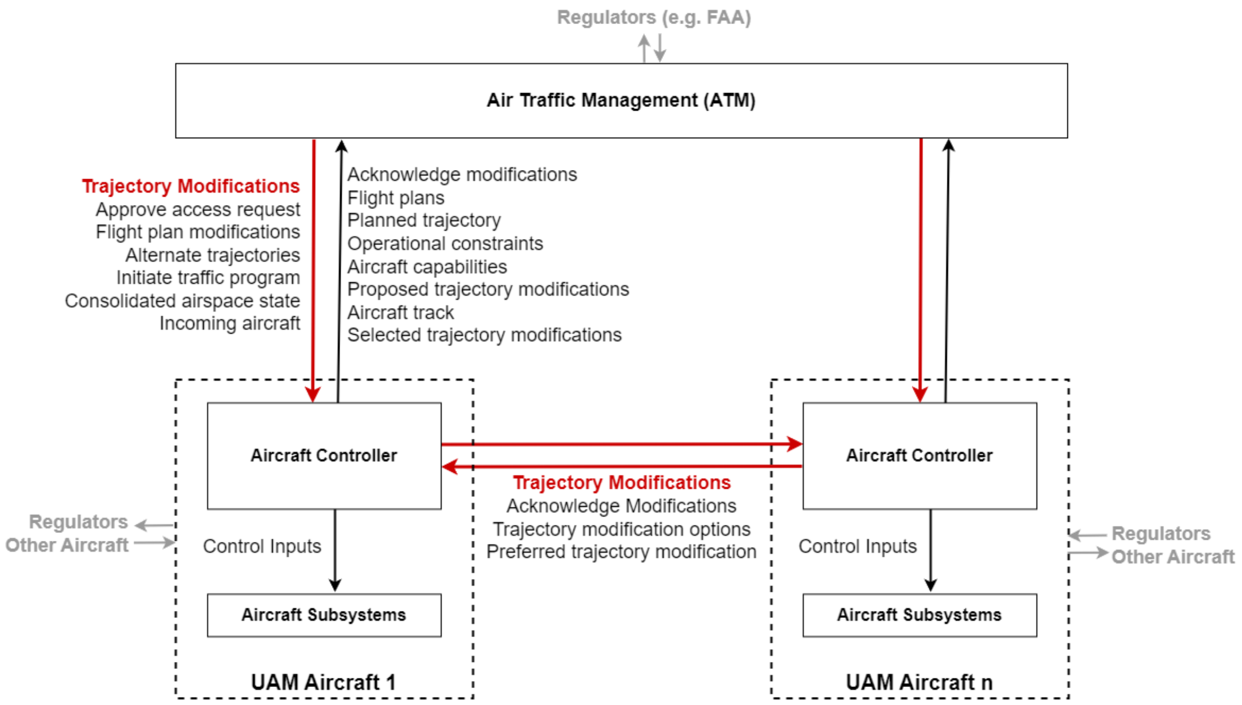


Figure 37: Zoomed-in control structure used in iteration 2 analyses

As shown in Figure 37, the zoomed-in control structure now models each aircraft as being comprised of an aircraft controller and the physical subsystems of the aircraft. This highlights the fact that to resolve a conflict in this shared collision avoidance architecture, the aircraft controller may either be given trajectory modifications selected by ATM or it may work with the controllers of other aircraft to select appropriate trajectory modifications for themselves. Regardless of how the trajectory modifications are selected, the aircraft controller eventually needs to decide what control inputs to provide to the aircraft subsystems to execute the trajectory modifications. Thus, by zooming in on the interactions between ATM and the aircraft, the control structure shown in Figure 37 better focuses the analysis on how the collective behavior of ATM and the aircraft may lead to inadequate resolution of conflicts.

Because this architecture requires ATM and the aircraft to work together to resolve conflicts, ATM and the aircraft are essentially collaborative controllers working as a team. Thus, STPA-Teaming [32] was used to analyze the *Trajectory Modifications* control action to determine how ATM and the aircraft collectively providing (or not providing) trajectory modifications could lead to unsafe behavior. The remainder of this section presents examples of the analysis results, and the full STPA-Teaming analysis can be found in Appendix E.

Consistent with the STPA-Teaming process [32], after creating the control structure to model the system, the next step is to generate unsafe collaborative control actions (UCCAs). As shown in Figure 37, either ATM or the aircraft can provide the *Trajectory Modifications* control action. Table 29 shows the three main combinations of this control action that were considered to identify type 1-2 abstracted UCCAs. The far-right column of Table 29 also describes the types of unsafe behaviors that are covered by each combination.

Table 29: Combinations of control actions considered to identify type 1-2 UCCAs

#	**Either ATM or the aircraft**	**While the other**	**Unsafe Behaviors Covered**
1	<u>Does not provide</u> Trajectory Modifications	<u>Does not provide</u> Trajectory Modifications	Identifies how a collision might remain unresolved by both ATM and the aircraft
2	<u>Provides</u> Trajectory Modifications	<u>Does not provide</u> Trajectory Modifications	Identifies how either ATM or the aircraft might provide unsafe trajectory modifications
3	<u>Does not provide</u> Trajectory Modifications	<u>Provides</u> Trajectory Modifications	*Duplicate - Same as #2*
4	<u>Provides</u> Trajectory Modifications	<u>Provides</u> Trajectory Modifications	Identifies how conflicts or inconsistencies in trajectory modifications might occur

For each of these combinations of control actions, multiple abstracted Type 1-2 UCCAs were identified and Table 30 shows examples of these UCCAs. Note that UCCA-17 is essentially the UCCA involved in the Uberlingen accident discussed in Section 5.1.

Table 30: Example Type 1-2 UCCAs for shared collision avoidance architecture

ID	Either ATM or one of the aircraft	While the others	Context
UCCA-1	Does not provide Trajectory Modifications	Does not provide Trajectory Modifications	when the trajectories of two aircraft are in conflict *[H-1]*
UCCA-3			when the trajectory of an aircraft conflicts with an obstacle or terrain *[H-1]*
UCCA-11	Provides Trajectory Modifications	Does not provide Trajectory Modifications	when the modifications will result in a secondary collision with another aircraft *[H-1]*
UCCA-14			when the modifications will cause a collision with an obstacle or terrain *[H-1]*
UCCA-17	Provides Trajectory Modifications	Provides Trajectory Modifications	when the trajectory modifications conflict with each other *[H-1]*

For UCCAs where neither ATM nor the aircraft provide trajectory modifications (e.g., UCCA-1 and UCCA-3), no further refinement of these UCCAs is needed because no additional detail is required to identify scenarios for those UCCAs. However, the other UCCAs can be refined and Table 31 shows examples of refined UCCAs for UCA-11 and UCCA-17.

Table 31: Example refined Type 1-2 UCCAs for shared collision avoidance architecture

ID	Sub-ID	ATM	Aircraft 1	Aircraft n	Context
UCCA-11	**UCCA-11.1**	Provides Trajectory Modifications	Does not provide Trajectory Modifications		when the modifications will result in a secondary collision with another aircraft *[H-1]*
	UCCA-11.2	Does not provide Trajectory Modifications	Provides Trajectory Modifications		
UCCA-17	**UCCA-17.1**	Provides Trajectory Modifications	Provides Trajectory Modifications	Does not provide Trajectory Modifications	when the trajectory modifications conflict with each other *[H-1]*
	UCCA-17.2	Does not provide Trajectory Modifications	Provides Trajectory Modifications	Provides Trajectory Modifications	
	UCCA-17.3	Provides Trajectory Modifications	Provides Trajectory Modifications	Provides Trajectory Modifications	

As shown in the first two rows of Table 31, UCCAs where either ATM or the aircraft (but not both) provide trajectory modifications can be refined to consider UCCAs where ATM provides the unsafe control action (and the aircraft do not) and UCCAs where the aircraft provide the unsafe control action (and ATM does not). In Table 31, UCCA-11.1 is an example of the former and UCCA-11.2 is an example of the latter.

Similarly, UCCA-17 can also be refined to consider three UCCAs:

1. ATM and one of the aircraft provide conflicting trajectory modifications (UCCA-17.1)
2. Two aircraft select conflicting trajectory modifications (UCCA-17.2)
3. ATM and the aircraft all select conflicting trajectory, some or all of which are conflicting (UCCA-17.3)

For Type 3-4 UCCAs, this analysis assumes that Trajectory Modifications is a discrete control action that simply indicates to aircraft what their new trajectory should be. Thus, only one type of Type 3-4 UCCA was considered: one controller provides trajectory modifications before the other controller provides trajectory modifications. Table 32 shows two examples of Type 3-4 UCCAs and Table 33 shows the three refined UCCAs identified for UCCA-18.

Table 32: Example Type 3-4 UCCAs for shared collision avoidance architecture

#	**Either ATM or the aircraft**	**Then the other**	**Context**
UCCA-18	Provides Trajectory Modifications	Provides Trajectory Modifications	when ATM and the aircraft are attempting to resolve the same conflict *[H-1]*
UCCA-19			When ATM and the aircraft are modifying trajectories for different reasons *[H-1]*

Table 33: Examples of refined type 3-4 UCCAs for UCCA-18

Sub-ID	**Trajectory Modifications provided by**	**Then trajectory Modifications provided by**	**Context**
UCCA-18.1	ATM	Aircraft n	when ATM and the aircraft are attempting to resolve the same conflict *[H-1]*
UCCA-18.2	Aircraft n	ATM	
UCCA-18.3	Aircraft 1	Aircraft n	

Once the UCCAs have been identified, causal scenarios can then be developed. To illustrate how this was done for the shared collision avoidance architecture, example scenarios for three UCCAs are presented in this section.

Example scenarios for UCCA-1

UCCA-1: Neither ATM nor the aircraft provide trajectory modifications when the trajectories of two aircraft are in conflict [H-1]

CS-2.1.1-1: The aircraft attempt to resolve the conflict and ATM does not. However, when the aircraft attempt to verify with ATM that their selected trajectory modifications will have alternate trajectory options available, ATM does not confirm this and therefore the aircraft are unable to select trajectory modifications before the collision occurs.

CS-2.1.1-2: ATM allows the aircraft to resolve the conflict. However, if the conflict involves a large number of aircraft or numerous operational constraints, it may take too long for the aircraft to coordinate among themselves to resolve the collision. As a result, neither ATM nor the aircraft issue trajectory modifications to prevent the conflict.

CS-2.1.1-4: ATM and the aircraft both assume the other is better equipped to resolve the conflict or they each wrongly believe the other will resolve the conflict. As a result, each waits for the other to resolve the conflict and neither of them selects trajectory modifications to prevent it.

Example scenarios for UCCA-11.1

UCCA-11.1: ATM provides trajectory modifications (and the aircraft do not) when the modifications will result in a collision [H-1]

CS-2.11.1.1-1: ATM selects trajectory modifications that contain secondary conflicts. It does this believing that it would be faster to issue these first and then resolve the secondary conflicts later. However, ATM becomes busy resolving other conflicts and does not return in time to resolve these secondary conflicts before a collision occurs. Furthermore, the aircraft believe ATM will resolve them and don't try to resolve them on their own.

CS-2.11.1.2-2: Other aircraft are about to but have not yet modified their trajectories and therefore ATM has not received any feedback that the trajectories of some aircraft are about to be modified when it begins to identify its own trajectory modifications. If it does not receive and process feedback later that the trajectories of some aircraft have been modified, ATM will identify trajectory modifications based on the outdated aircraft trajectories and select trajectory modifications that it does not realize are in conflict with the updated trajectories of some aircraft.

CS-2.11.1.3-1: ATM correctly selects trajectory modifications that will not result in a collision. However, during transmission to the aircraft, some of the trajectory modifications are dropped (e.g., due to a communications error) and only some of the aircraft receive trajectory modifications. As a result, aircraft that did receive trajectory modifications may now conflict with those that did not receive trajectory modifications. However, since the aircraft believe that ATM is managing the collision, they do not check these modifications and simply execute them.

Example scenarios for UCCA-17.1

UCCA-17.1: ATM and the aircraft both provide trajectory modifications when they conflict with each other [H-1]

CS-2.17.1.1-1: Both ATM and a UAM aircraft identify a potential conflict with another aircraft that is not equipped to perform self-separation. The UAM aircraft proceeds to resolve the conflict under the assumption that the other aircraft will not change trajectory and are able to identify a solution first. However, ATM can control that other aircraft (e.g., by coordinating with conventional ATC). Thus, although ATM knows the aircraft has already selected trajectory modifications, ATM provides what it believes to be a better solution but that conflicts with those of the aircraft.

CS-2.17.1.1-6: Multiple sets of conflicts are occurring and ATM and the aircraft have received feedback about them and are attempting to resolve them. While the aircraft are each only attempting to resolve their own local conflict, ATM is resolving all these conflicts together because it believes it can resolve them more efficiently. As a result, although the aircraft have selected trajectory modifications already, ATM provides a conflicting set to the aircraft.

CS-2.17.1.2-1: Both ATM and the aircraft identify a potential conflict and attempt to resolve it. If they both select trajectory modifications at about the same time, neither ATM nor the aircraft will receive feedback that the other has already selected trajectory modifications before they provide their own. Thus, they both provide trajectory modifications that conflict with each other.

5.3 Developing the Shared Collision Avoidance Conceptual Architecture

Having analyzed the shared collision avoidance architecture using STPA-Teaming, the identified scenarios can be used to refine the conceptual architecture created in design iteration 1 to define the control elements that are needed to adequately manage shared collision avoidance. Because this design iteration is focused on deciding how to implement shared collision avoidance, Resp-1 (the responsibility for resolving conflicts) will be refined in this design iteration to determine the more detailed sub-responsibilities that will need to be performed. Figure 38 shows a refocused version of the conceptual architecture created in design iteration 1 that highlights Resp-1 that will be refined.

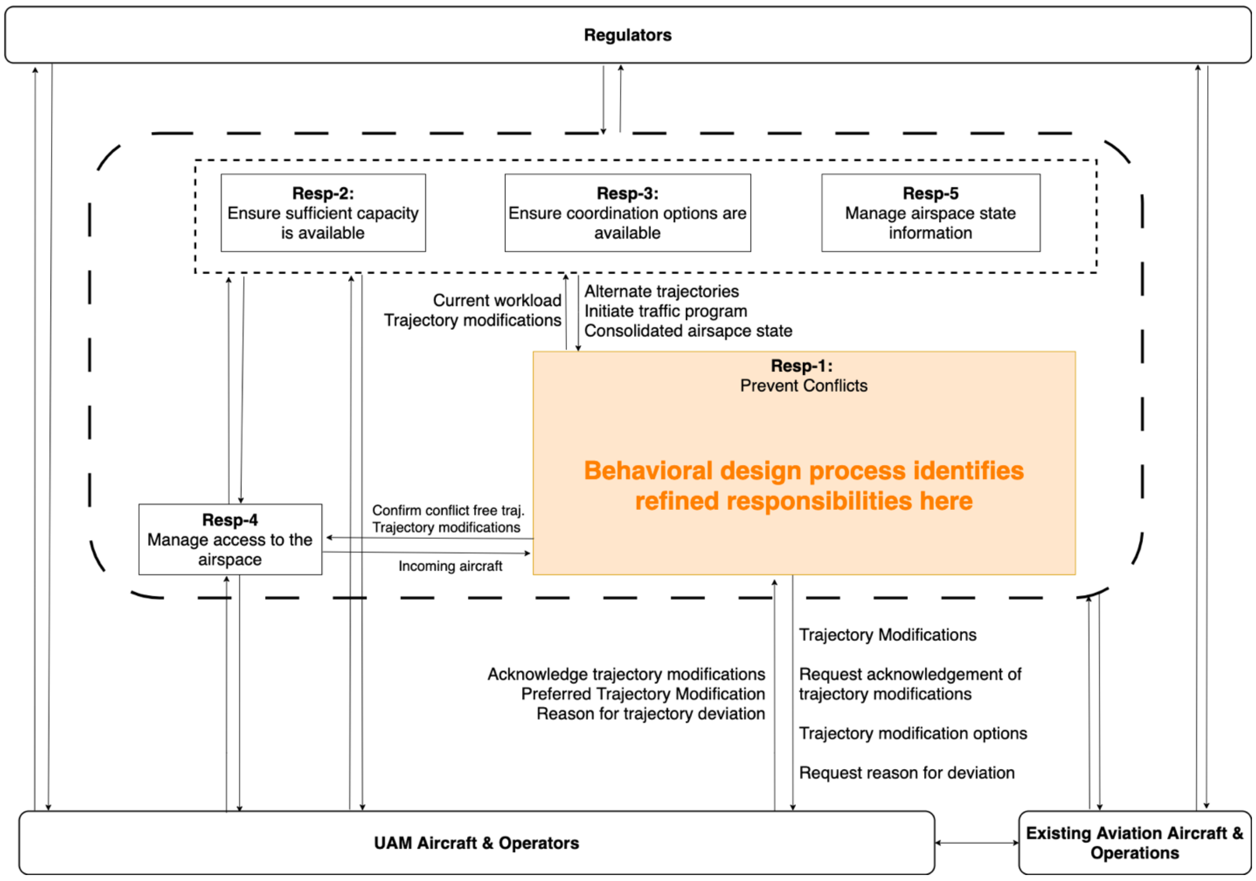

Regulators
Resp-2:
Ensure sufficient capacity is available
Resp-3:
Ensure coordination options are available
Resp-5
Manage airspace state information
Current workload
Trajectory modifications
Alternate trajectories
Initiate traffic program
Consolidated airsapce state
Resp-1:
Prevent Conflicts
Behavioral design process identifies refined responsibilities here
Resp-4
Manage access to the airspace
Confirm conflict free traj.
Trajectory modifications
Incoming aircraft
Trajectory Modifications
Acknowledge trajectory modifications
Preferred Trajectory Modification
Reason for trajectory deviation
Request acknowledgement of trajectory modifications
Trajectory modification options
Request reason for deviation
UAM Aircraft & Operators
Existing Aviation Aircraft & Operations

Figure 38: Iteration 1 conceptual architecture with Resp-1 highlighted for refinement

This section describes how this refinement of Resp-1 is done. First, additional system requirements are generated that define the additional safety constraints necessary to ensure safe shared collision avoidance. The augmented set of system requirements are then used to define a refined set of sub-responsibilities of Resp-1 and the process model parts, control actions and feedback that are needed for each sub-responsibility. Finally, the sub-responsibilities are assembled to create the refined conceptual architecture.

5.3.1 Additional System Requirements for Shared Collision Avoidance

Based on the scenarios identified using STPA-Teaming, additional system requirements can be generated that describe the additional safety constraints needed to ensure safe shared collision avoidance. Table 34 shows three examples of how requirements are derived from the scenarios and Table 35 shows several more examples of system requirements generated to prevent unsafe behavior arising from shared collision avoidance. The full set of additional system requirements generated in this design iteration are listed in Appendix F.

Note that CS-2.17.1.2-1 (second row of Table 34) is essentially the scenario that occurs in the Uberlingen accident. Thus, by identifying that scenario now during development, the necessary system requirements can be generated and the right control elements can be designed into the ATM architecture to prevent an accident like the Uberlingen accident from occurring.

Table 34: Examples of derived system requirements for shared collision avoidance

Scenario	Derived Requirement
CS-2.1.1-4: ATM and the aircraft both assume the other is better equipped to resolve the conflict or they each wrongly believe the other will resolve the conflict. As a result, each waits for the other to resolve the conflict and neither of them selects trajectory modifications to prevent the conflict.	**Req-121:** An explicit decision must be made about who is resolving a potential conflict
CS-2.17.1.2-1: Both ATM and the aircraft identify a potential conflict and attempt to resolve it. If they both select trajectory modifications at the same time, neither receives feedback that the other has already selected trajectory modifications before they select their own. Thus, they provide trajectory modifications that conflict with each other.	**Req-123:** If multiple potential resolutions to a conflict are identified, an explicit decision must be made about which trajectory modifications to execute
CS-2.17.1.1-6: Multiple conflicts occur, and ATM and the aircraft are attempting to resolve them. While the aircraft are each only attempting to resolve their own local conflict, ATM is resolving all these conflicts together because it believes it can resolve them more efficiently. As a result, although the aircraft have selected trajectory modifications already, ATM provides a conflicting set to the aircraft.	**Req-124:** Under <TBD> conditions, to better coordinate the resolution of conflicts, it must be possible to temporarily require that all trajectory modification decisions be made centrally.

Table 35: Additional examples of system requirements for shared collision avoidance

ID	Requirement
Req-103	If either ATM or the aircraft is unable to resolve a potential conflict, the other must be able to take over and resolve it.
Req-104	Conflicts must continue to be resolved even if the ability of ATM or one aircraft to do so is compromised.
Req-108	Any aircraft within <TBD> distance of an identified conflict must be included in coordination to ensure secondary collisions are avoided.
Req-136	Trajectory modification decisions must account for aircraft changing trajectories
Req-144	Air traffic priorities must be determined and adhered to consistently when making trajectory modification decisions
Req-145	The ability of aircraft to execute their planned trajectory to the required navigational performance must be monitored and modifications reconsidered if they are unable to execute their planned trajectories sufficiently accurately

5.3.2 Creating the Refined Conceptual Architecture for Shared Collision Avoidance

Once these additional system requirements have been generated, refined responsibilities can then be identified. To do this, the additional requirements derived from the STPA-Teaming results are combined with the original requirements used to define Resp-1 to create an expanded set of requirements.

From this expanded set of requirements, new requirements groups are formed. As was done in iteration 1, each requirement group is defined by a control requirement and the constraint requirements that apply to it. The difference in this iteration is that the control requirements are more detailed than those in iteration 1. More detailed control responsibilities and associated responsibility constraints can then be generated for each group.

Table 36 shows an example of one requirement group and the refined responsibility and constraints that were derived. Table 37 summarizes the six responsibilities that were identified in this iteration to refine Resp-1. The full set of control actions and feedback for all six responsibilities in this design iteration are shown in Appendix F.

Table 36: Example derivation of refined responsibility and associated constraints

Requirements Group
Req-121: An explicit decision must be made about who is resolving a potential conflict
Req-103: If either ATM or the aircraft is unable to resolve a potential conflict, the other must be able to take over and resolve it.
Req-104: Conflicts must continue to be resolved even if the ability of ATM or one aircraft to do so is compromised.
Req-108: Any aircraft within <TBD> distance of an identified conflict must be included in coordination to ensure secondary collisions are avoided.
Req-109: A potential conflict that remains unresolved after <TBD> of being identified must be prioritized and resolved within <TBD> time.
Control Responsibility (Resp) and Associated Constraints (RC)
Resp-1.2: Decide which controller is resolving a conflict *[Req-121]*
RC-78: Allow ATM and the aircraft to take over from each other to resolve a conflict *[Req-103]*
RC-79: Continue preventing conflicts even if the ability of ATM or one of the aircraft to do so is compromised [*Req-104]*
RC-83: Aircraft within <TBD> of an area where a potential conflict might occur should be included in coordination *[Req-108]*
RC-84: Prioritize and resolve any conflict within <TBD> time that remains unresolved after <TBD> of being identified *[Req-109]*

Table 37: The six refined sub-responsibilities of Resp-1

Control Requirement	Control Responsibility
Req-4*: ATM system shall identify and resolve any conflict with an aircraft's trajectory including conflicts between two aircraft trajectories or a conflict of an aircraft trajectory with terrain	**Resp-1.1**: Identify and resolve potential conflicts
Req-121: There must be an explicit decision on who will be resolving a conflict	**Resp-1.2:** Decide which controller is resolving a conflict
Req-123: If multiple potential resolutions to a conflict are identified, an explicit decision must be made about which trajectory modification instructions to execute	**Resp-1.3:** Arbitrate conflicting conflict resolution proposals
Req-145: The ability of aircraft to execute their planned trajectory to the required navigational performance must be monitored and trajectory modifications reconsidered if they are unable to execute their planned trajectories sufficiently accurately	**Resp-1.4:** Ensure aircraft are adequately following their planned trajectory and any modifications made to it
Req-144: Air traffic priorities must be determined and adhered to consistently when making trajectory modification decisions	**Resp-1.5:** Prioritize air traffic to inform trajectory modification decisions
Req-124: Under <TBD> conditions, to better coordinate the resolution of conflicts, it must be possible to temporarily require that all trajectory modification decisions be made centrally.	**Resp-1.6:** Establish when trajectory modification decisions need to be made centrally

** Note: In iteration 1, Req-4 was used to derive Resp-1. In iteration 2, it is used again to derive the more detailed responsibility Resp-1.1*

At this point, it is worth noting that this process has identified control responsibilities that could help to prevent accidents like the Uberlingen accident from occurring if they were implemented in the ATC system architecture. For example, consider Resp-1.2. As discussed in Section 5.1, the ATC system that existed at the time of the Uberlingen accident was designed such that TCAS and ATC operated independently with no direct communication between them. Thus, a responsibility like Resp-1.2 was not included because federal regulations and FAA guidance prescribe that ATC and TCAS are both responsible for resolving any conflict that they identify. Even in today's ATC system, a responsibility like Resp-1.2 does not exist even though it could help to prevent unsafe independent conflict resolution decisions being made by ATC and TCAS by clarifying which of them will make the resolution decision for a given conflict.

For this shared collision avoidance architecture, however, Resp-1.2 is critically needed because it will enable the active decision making necessary to allow the ATM system to adapt its collision avoidance behavior to the prevailing air traffic circumstances. Thus, this architecture development framework can help to design the behavior of this responsibility and decide how to incorporate it into the ATM architecture.

Having defined these six refined responsibilities, the required process model parts, control actions and feedback as well as the control action targets and feedback sources for each refined responsibility can then be defined using the same process that was used in iteration 1 (and described in Section 3.3). Table 38 shows an example of how these control elements were defined for Resp-1.2 to enable it to adequately manage and coordinate the conflict resolution efforts of ATM and the aircraft. Each control element is traced to the responsibility or constraint that it was derived from to record the rationale for including each control element. The control elements for all six refined responsibilities can be found in Appendix F.

Table 38: Identifying process model parts, control actions and feedback for Resp-1.2

Resp-1.2: Decide which controller is resolving a conflict **RC-78:** Allow ATM and the aircraft to take over from each other to resolve a conflict **RC-79:** Continue preventing conflicts even if the ability of ATM or one of the aircraft to do so is compromised **RC-83:** Aircraft within <TBD> of an area where a potential conflict might occur should be included in coordination **RC-84:** Prioritize and resolve any conflict within <TBD> time that remains unresolved after <TBD> of being identified	
Summary of Desired Behavior	When a potential conflict is identified, consider the conditions under which the conflict will occur and decide if it would be more easily resolved by ATM or the aircraft. Once a decision is made, inform both ATM and the aircraft of who is assigned to resolve the conflict. The conditions might include: 1. Current workload of ATM and the aircraft 2. Traffic density and number of aircraft involved in the conflict 3. Urgency of the conflict
Process Model Parts & Required Feedback/Inputs	• Feedback from Resp-1.1: o Identified conflicts *[Resp-1.2]* o Requested controller to resolve a conflict *[RC-78, RC-85]* o Unable to resolve conflict *[RC-78, RC-79]* • Input from Resp-4: Anticipated airspace state *[Resp-1.2, RC-83]* • Feedback from Resp-1.4: Unresolved collisions *[RC-79, RC-84]*
Required Control Actions/Outputs	• Control actions to Resp-1.1: o Assigned controller resolving conflict *[Resp-1.2]* o Aircraft involved in conflict *[RC-83]*

Once the required control elements are defined for all the refined responsibilities, they can be assembled to create the refined shared collision avoidance conceptual architecture shown in Figure 39.

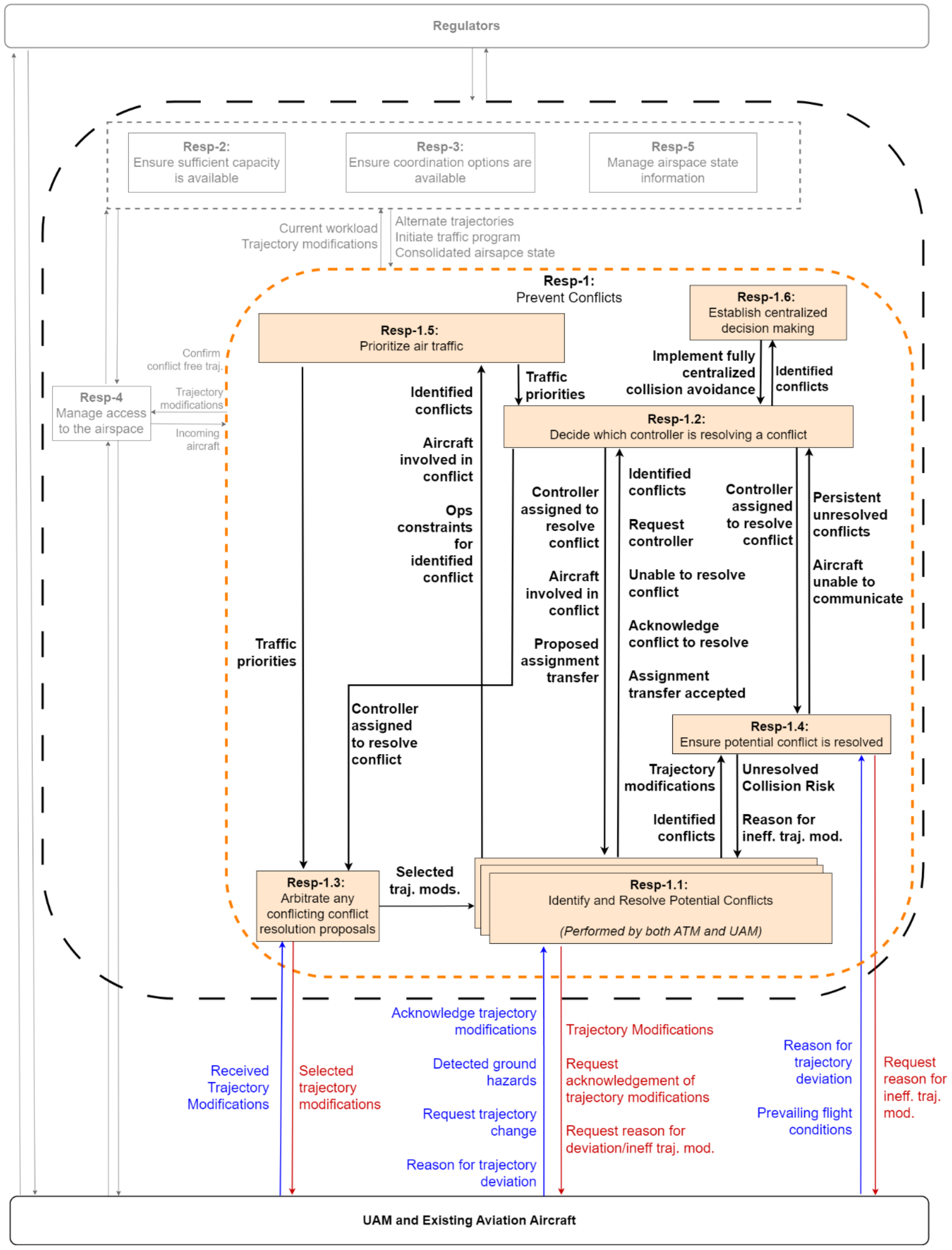

Regulators
Resp-2: Ensure sufficient capacity is available
Resp-3: Ensure coordination options are available
Resp-5 Manage airspace state information
Current workload
Trajectory modifications
Alternate trajectories
Initiate traffic program
Consolidated airsapce state
Resp-1: Prevent Conflicts
Resp-1.6: Establish centralized decision making
Resp-1.5: Prioritize air traffic
Confirm conflict free traj.
Resp-4 Manage access to the airspace
Trajectory modifications
Incoming aircraft
Identified conflicts
Aircraft involved in conflict
Ops constraints for identified conflict
Traffic priorities
Implement fully centralized collision avoidance
Identified conflicts
Resp-1.2: Decide which controller is resolving a conflict
Controller assigned to resolve conflict
Aircraft involved in conflict
Proposed assignment transfer
Identified conflicts
Request controller
Unable to resolve conflict
Acknowledge conflict to resolve
Assignment transfer accepted
Controller assigned to resolve conflict
Persistent unresolved conflicts
Aircraft unable to communicate
Traffic priorities
Controller assigned to resolve conflict
Resp-1.4: Ensure potential conflict is resolved
Trajectory modifications
Identified conflicts
Unresolved Collision Risk
Reason for ineff. traj. mod.
Resp-1.3: Arbitrate any conflicting conflict resolution proposals
Selected traj. mods.
Resp-1.1: Identify and Resolve Potential Conflicts
(Performed by both ATM and UAM)
Received Trajectory Modifications
Selected trajectory modifications
Acknowledge trajectory modifications
Detected ground hazards
Request trajectory change
Reason for trajectory deviation
Trajectory Modifications
Request acknowledgement of trajectory modifications
Request reason for deviation/ineff traj. mod.
Reason for trajectory deviation
Prevailing flight conditions
Request reason for ineff. traj. mod.
UAM and Existing Aviation Aircraft

Figure 39: Refined shared collision avoidance conceptual architecture

As shown in Figure 39, this refined conceptual architecture still contains all the responsibilities and interactions that were defined in the conceptual architecture that was created in the first design iteration. Because of the design decisions made in this design iteration, Resp-1 (the responsibility for resolving conflicts) has now been refined into six more detailed responsibilities with a more detailed set of control actions and feedback between them.

Per the decision made in design iteration 1, both ATM and UAM are responsible for identifying and resolving conflicts (Resp-1.1). Therefore, Resp-1.1 is depicted in Figure 39 with multiple boxes to represent that there are multiple copies of this responsibility in the conceptual architecture. Above that are the other five sub-responsibilities that control the various aspects of when and how Resp-1.1 is carried out. These responsibilities include deciding who should resolve an identified conflict (Resp-1.2), arbitrating conflicting trajectory modifications (Resp-1.3) and ensuring that potential conflicts are resolved or re-evaluated if they remain unresolved (Resp-1.4). In addition, above Resp-1.2 are two higher-level traffic management responsibilities. Resp-1.5 prioritizes the air traffic involved in a conflict to ensure that air traffic is prioritized appropriately, and those priorities are followed consistently when resolving conflicts, regardless of whether ATM or the aircraft resolve the conflict. Resp-1.6, then, monitors the state of the airspace and the collisions that are occurring to decide when it might be beneficial to temporarily resolve all conflicts centrally to minimize the amount of coordination required before a conflict is resolved. Resp-1.6 can then implement this centralized decision making "mode" using its control action to Resp-1.2.

Figure 39 therefore shows that this behavioral design process enables systems engineers to refine a conceptual architecture and define the control behavior needed to prevent accidents like the Uberlingen accident from occurring. For example, responsibility Resp-1.2 (discussed earlier in this section) now has its control actions and feedback defined and the conceptual architecture shows how its behavior, in conjunction with other responsibilities, ensures that the efforts of ATM and the aircraft to identify and resolve conflicts are adequately coordinated.

5.4 Comparing Architecture Options for Implementing Shared Collision Avoidance

Having defined this conceptual architecture for shared collision avoidance, it can then be analyzed using STPA to generate scenarios that can inform what architecture options should be considered to implement it. This section discusses how the STPA results were used to create architecture options for comparison and the full STPA analysis of the refined conceptual architecture can be found in Appendix G.

Of the six refined responsibilities, three of them have relatively straightforward assignments. Per the decision made in iteration 1, Resp-1.1 should be assigned to both ATM and the aircraft since this shared collision avoidance architecture intends for both ATM and the aircraft to be able to resolve conflicts. For Resp-1.5 and Resp-1.6, because they essentially involve making higher-level traffic management decisions such as prioritizing air traffic and deciding when conflicts should all be resolved centrally, it is likely that assigning them to ATM will be the best option (of course, other assignment options could be explored if desired).

However, it is less clear if Resp-1.2, Resp-1.3, and Resp-1.4 should be assigned to ATM or the aircraft. Given the relevance of Resp-1.2 to the Uberlingen accident, this research therefore focuses on using the results from the STPA analysis of the refined conceptual architecture to explore different architecture options for assigning Resp-1.2.

5.4.1 Identifying Assignment Constraints and Creating Architecture Options

Based on the STPA analysis of the refined conceptual architecture, several scenarios suggested that there could be potential benefits to assigning Resp-1.2 to either ATM or the aircraft. Table 39 shows examples of these scenarios, the assignment constraints derived from them, and the reason for that preferred assignment.

Table 39: Examples of assignment constraints derived from STPA scenarios

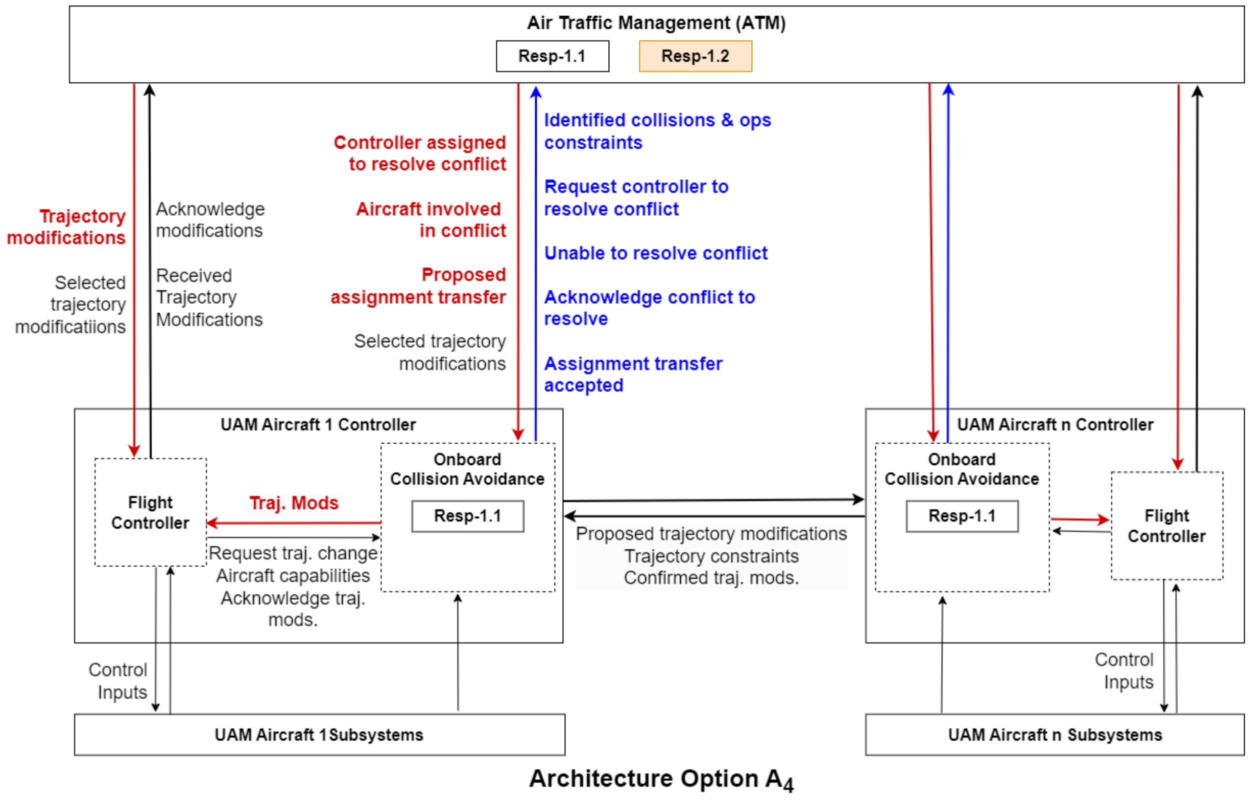
Scenario	Assignment Constraint	Reason for Assignment Constraint
CS-3.1.1-5: Neither ATM nor the aircraft are assigned to resolve a conflict. This could occur if the aircraft identify an urgent conflict but need to wait for a decision on who should resolve the conflict. By the time they receive that decision, there is not enough time to select trajectory modifications before the conflict occurs.	Resp-1.2 = Aircraft	This assignment ensures that the aircraft can respond to urgent conflicts quickly when they identify them
CS-3.11.1.1-2: ATM is inappropriately assigned to resolve a conflict based on outdated feedback about flight conditions or aircraft capabilities. It therefore selects trajectory modifications that the aircraft cannot adequately execute and a collision occurs.		The aircraft have more timely access to feedback about flight conditions or aircraft capabilities to inform decision making
CS-3.1.2-1: ATM is inappropriately assigned to resolve a conflict even though it is already under a high workload. Thus, the additional conflict assignment exceeds ATM's capabilities and it is unable to make an adequate trajectory modification decision in time.	Resp-1.2 = ATM	It is easier for ATM to know its own workload than for the aircraft to estimate that
CS-3.1.2-4: The aircraft are assigned to resolve a conflict based on the urgency of the conflict but using outdated information about its context. However, at least one of them is in a critical phase of flight (high workload) and they do not select trajectory modifications before a collision occurs.		ATM has broader situational awareness of the state of the airspace and the trajectory constraints for each aircraft

The preferred assignments in Table 39 suggest that there are potential benefits to assigning Resp-1.2 to either ATM or the aircraft as described by the reason for each assignment constraint. These two assignments are therefore worth exploring to determine the benefits and tradeoffs between them. For this reason, this design iteration explores these two potential architecture options for implementing Resp-1.2 in the ATM architecture.

Table 40 provides an overview of how each of the six refined responsibilities are assigned in these two architecture options. As in design iteration 1, only one responsibility's assignment is changed (Resp-1.2). The assignments for the other responsibilities are kept the same for both architecture options. So, Resp-1.1 is assigned to both ATM and the aircraft (consistent with the decision in design iteration 1) and Resp-1.3, Resp-1.4, Resp-1.5, and Resp-1.6 are all assigned to ATM only. Simplified control structures for each of these two architecture options are shown in Figure 40 and Figure 41.

Table 40: Two architecture options for assigning Resp-1.2

Resp. ID	**Responsibility**	**Option A$_4$** Centralized Allocation of Conflicts	**Option A$_5$** Airborne Allocation of Conflicts
Resp-1.1	Identify and resolve potential conflicts	ATM & Aircraft	ATM & Aircraft
Resp-1.2	Decide which controller is resolving a conflict	**ATM**	**Aircraft**
Resp-1.3	Arbitrate any conflicting conflict resolution proposals	ATM	ATM
Resp-1.4	Ensure identified conflicts are resolved	ATM	ATM
Resp-1.5	Prioritize air traffic	ATM	ATM
Resp-1.6	Establish centralized conflict resolution	ATM	ATM



Air Traffic Management (ATM)
Resp-1.1
Resp-1.2
Trajectory modifications
Selected trajectory modificatiions
Acknowledge modifications
Received Trajectory Modifications
Controller assigned to resolve conflict
Aircraft involved in conflict
Proposed assignment transfer
Selected trajectory modifications
Identified collisions & ops constraints
Request controller to resolve conflict
Unable to resolve conflict
Acknowledge conflict to resolve
Assignment transfer accepted
UAM Aircraft 1 Controller
Flight Controller
Traj. Mods
Onboard Collision Avoidance
Resp-1.1
Request traj. change
Aircraft capabilities
Acknowledge traj. mods.
Proposed trajectory modifications
Trajectory constraints
Confirmed traj. mods.
UAM Aircraft n Controller
Onboard Collision Avoidance
Resp-1.1
Flight Controller
Control Inputs
Control Inputs
UAM Aircraft 1 Subsystems
UAM Aircraft n Subsystems
Architecture Option A4

Figure 40: Architecture option A$_4$ with Resp-1.2 control actions and feedback highlighted

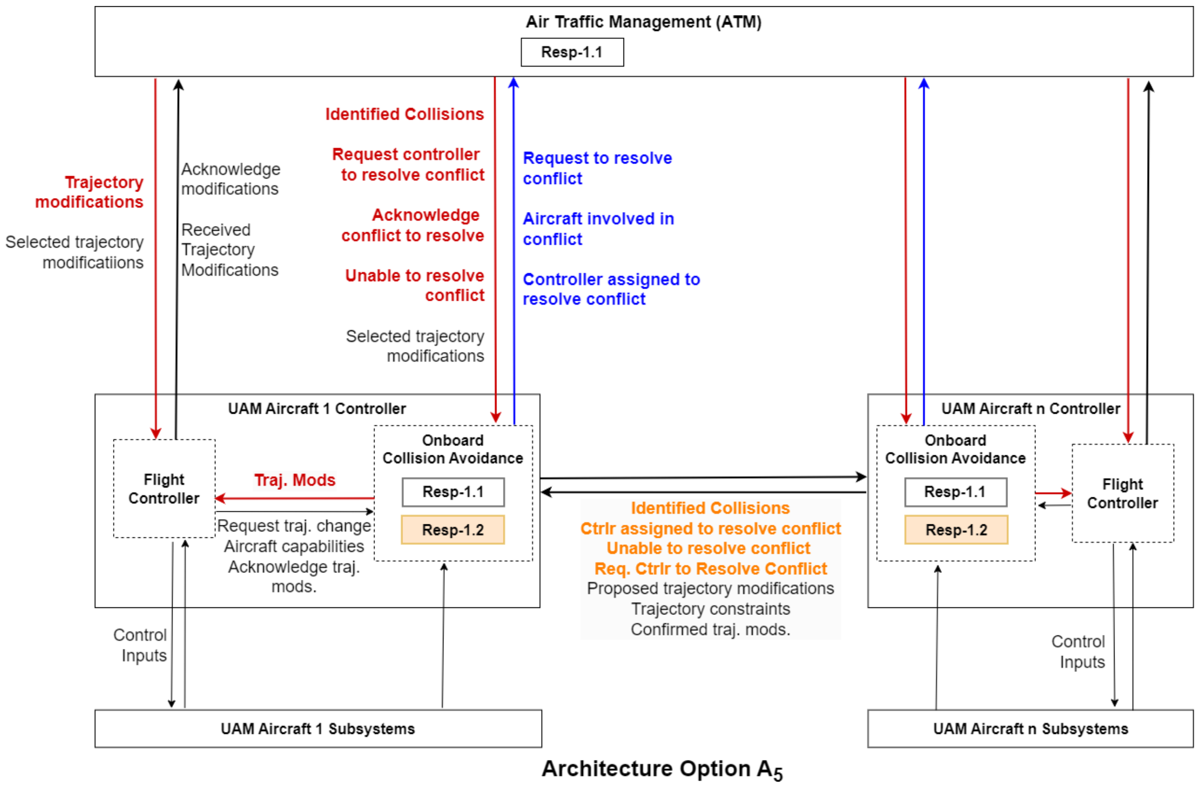

Air Traffic Management (ATM)
Resp-1.1
Trajectory modifications
Selected trajectory modificatiions
Acknowledge modifications
Received Trajectory Modifications
Identified Collisions
Request controller to resolve conflict
Acknowledge conflict to resolve
Unable to resolve conflict
Selected trajectory modifications
Request to resolve conflict
Aircraft involved in conflict
Controller assigned to resolve conflict
UAM Aircraft 1 Controller
Flight Controller
Traj. Mods
Onboard Collision Avoidance
Resp-1.1
Resp-1.2
Request traj. change
Aircraft capabilities
Acknowledge traj. mods.
Identified Collisions
Ctrlr assigned to resolve conflict
Unable to resolve conflict
Req. Ctrlr to Resolve Conflict
Proposed trajectory modifications
Trajectory constraints
Confirmed traj. mods.
UAM Aircraft n Controller
Onboard Collision Avoidance
Resp-1.1
Resp-1.2
Flight Controller
Control Inputs
Control Inputs
UAM Aircraft 1 Subsystems
UAM Aircraft n Subsystems
Architecture Option A5

Figure 41: Architecture option A_5 with Resp-1.2 control actions and feedback highlighted

As shown in Figure 40, architecture option A_4 assigns Resp-1.2 to ATM. Thus, in this architecture option, the aircraft can provide feedback to ATM of any identified conflicts. ATM can then decide to resolve the conflict itself or assign it to the aircraft to resolve. However, the aircraft do not resolve a conflict until ATM assigns the conflict to them.

By contrast, architecture option A_5 (Figure 41) assigns Resp-1.2 to the aircraft, thus essentially inverting the relationship between ATM and the aircraft when performing Resp-1.2. In this architecture option, ATM can indicate to the aircraft any identified conflicts. The aircraft can then decide to resolve the conflict themselves or request ATM's assistance to resolve it. However, ATM does not resolve a conflict until an aircraft makes the request for it to do so.

5.4.2 Evaluating and Comparing Architecture Options

As was done in iteration 1, STPA can then be used to analyze and compare these two architecture options. Table 43 shows two example scenarios that illustrate two tradeoffs that were identified. The remainder of this section discusses several other benefits and tradeoffs that were identified from this comparison. The full comparison table showing all the STPA scenarios that were used to compare these two architecture options can be found in Appendix H.

Table 41: Architecture comparison table for four example scenarios

#	Scenario	Scenario Occurs?		Evaluation Criteria
		A_4	A_5	
1	ATM identifies an urgent conflict that needs to be resolved. However, <controller performing Resp-1.2> takes too long to decide who should resolve a conflict. By the time ATM receives that decision, there is not enough time to select trajectory modifications before the conflict occurs	No	Yes	**Responsiveness** of trajectory modification decisions to prevent loss of separation when ATM resolves an urgent conflict
2	The aircraft identify an urgent conflict that needs to be resolved. However, <controller performing Resp-1.2> takes too long to decide who should resolve a conflict. By the time the aircraft receive that decision, there is not enough time to select trajectory modifications before the conflict occurs	Yes	No	**Responsiveness** of trajectory modification decisions to prevent loss of separation when the aircraft resolves an urgent conflict

The two scenarios in Table 41 show that the assignment of Resp-1.2 changes the ability of ATM or the aircraft to make timely trajectory modifications decisions to resolve a conflict. This observation can be made by comparing how architecture options A_4 and A_5 behave in each of these two scenarios. Figure 42 illustrates their behavior in scenario 1 and Figure 43 illustrates their behavior in scenario 2.

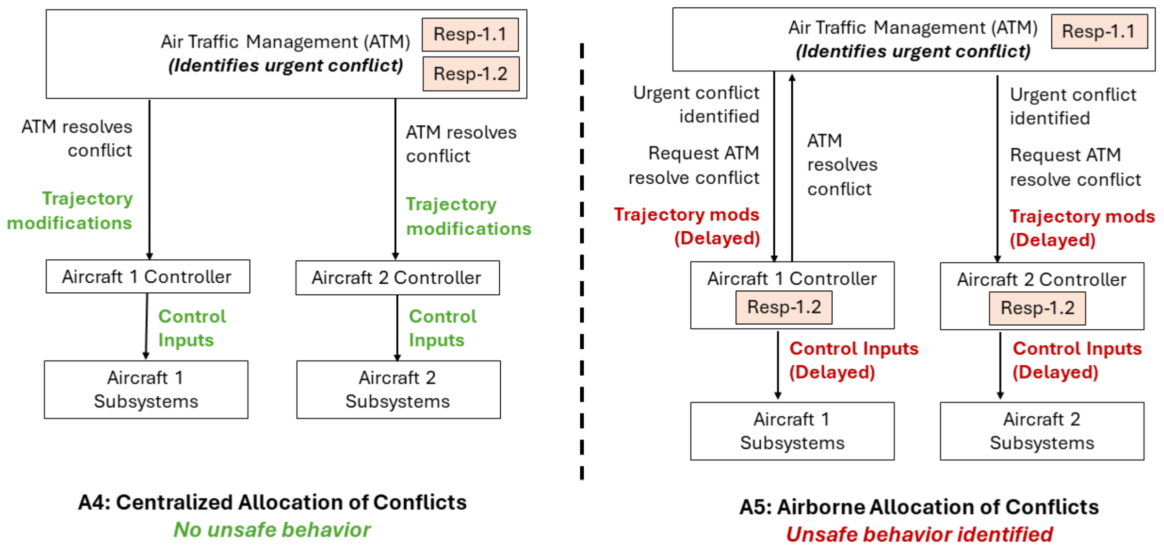


Figure 42: Behavior of A_4 (left) and A_5 (right) in scenario 1 in Table 41

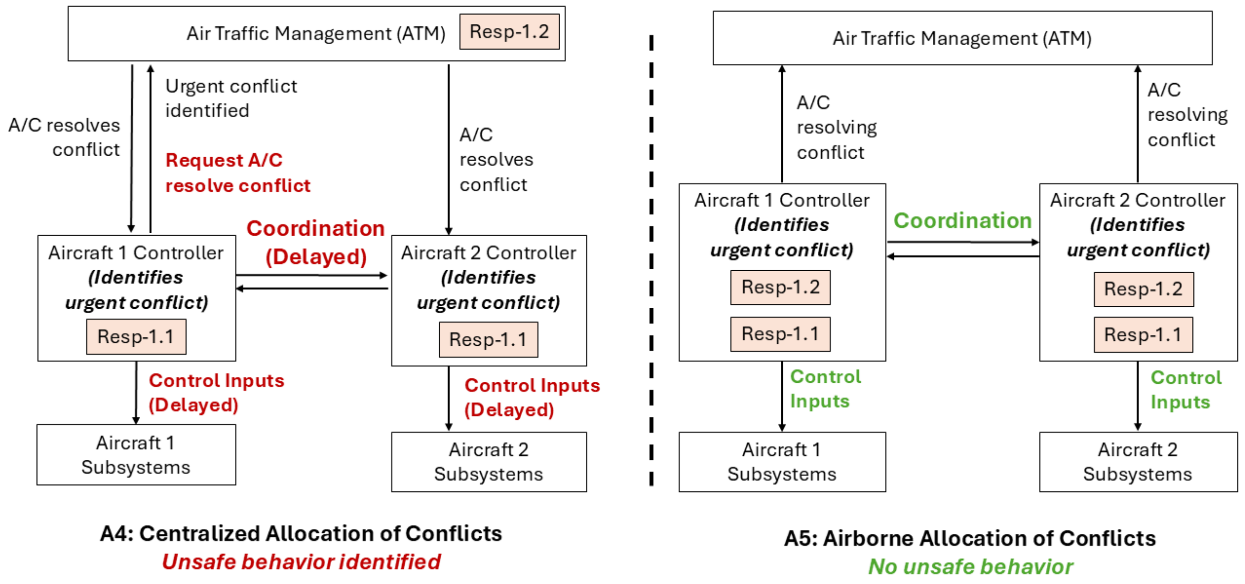

Air Traffic Management (ATM)
Resp-1.2
A/C resolves conflict
Urgent conflict identified
Request A/C resolve conflict
A/C resolves conflict
Coordination (Delayed)
Aircraft 1 Controller (Identifies urgent conflict)
Resp-1.1
Aircraft 2 Controller (Identifies urgent conflict)
Resp-1.1
Control Inputs (Delayed)
Control Inputs (Delayed)
Aircraft 1 Subsystems
Aircraft 2 Subsystems
A4: Centralized Allocation of Conflicts
Unsafe behavior identified
Air Traffic Management (ATM)
A/C resolving conflict
A/C resolving conflict
Aircraft 1 Controller (Identifies urgent conflict)
Coordination
Aircraft 2 Controller (Identifies urgent conflict)
Resp-1.2
Resp-1.1
Resp-1.2
Resp-1.1
Control Inputs
Control Inputs
Aircraft 1 Subsystems
Aircraft 2 Subsystems
A5: Airborne Allocation of Conflicts
No unsafe behavior

Figure 43: Behavior of A4 (left) and A5 (right) in scenario 2 in Table 41

In both scenarios, either ATM or the aircraft identifies and then resolves an urgent conflict. The difference between the two scenarios is which of them identifies the conflict and how quickly they can resolve it. In scenario 1 (Figure 42), ATM identifies the conflict and wants to resolve it. For this scenario, no unsafe behavior is observed for option A_4 because Resp-1.2 is assigned to ATM. Thus, ATM can immediately begin resolving the conflict as soon as they identify it. By contrast, in option A_5, Resp-1.2 is assigned to the aircraft and therefore ATM cannot resolve the conflict as soon as it identifies it. This is because, as designed in the conceptual architecture, a decision must be made about who should resolve the conflict before any controller can start resolving a conflict. Thus, ATM must first indicate the conflict to the aircraft and then wait for the aircraft to allocate the conflict to ATM. Only once the aircraft provide feedback requesting ATM to resolve the conflict can ATM start resolving it. This results in a delay in ATM selecting trajectory modifications and therefore a delay before the aircraft can execute those trajectory modifications. Depending on the urgency of the conflict, that delay could be large enough that the conflict is not resolved before a collision occurs.

Comparing the behavior of architecture options A_4 and A_5 in this first scenario shows that the main behavioral difference between them is the responsiveness (i.e., timeliness) with which ATM can resolve a conflict. Specifically, architecture option A_4 enables more responsive trajectory modification decisions by ATM than A_5. This therefore leads to the formulation of the evaluation criterion for scenario 1 as shown in Table 41.

The opposite behavior is seen when comparing the behavior of both architecture options in scenario 2 (Figure 43). In this scenario, the aircraft identify and want to resolve an urgent conflict. However, in option A_4, because Resp-1.2 is assigned to ATM, the aircraft must first provide feedback to ATM about the conflict and then wait for ATM to assign the conflict to them to resolve. This results in a delay before the aircraft can coordinate to select trajectory modifications and therefore a delay before they can maneuver to resolve the conflict. As in scenario 1, depending on the urgency of the conflict, that delay could be large enough that the aircraft cannot resolve the conflict in time before a collision occurs. By contrast, in option A_5, because

Resp-1.2 is assigned to the aircraft, the aircraft can resolve a conflict as soon as they identify it. Thus, no unsafe behavior occurs for option A_5.

Comparing the behavior of architecture options A_4 and A_5 in this second scenario, the main behavioral difference between them is the responsiveness with which the aircraft (instead of ATM) can resolve a conflict. Specifically, architecture option A_5 enables more responsive trajectory modification decisions by the aircraft than A_4. This therefore leads to the formulation of the evaluation criterion for scenario 2 as shown in Table 41.

These differences in behavior therefore show that the benefit of option A_4 is that ATM is more responsive (i.e., timely) in resolving an urgent conflict whereas the benefit of A_5 is that the aircraft are more responsive in resolving an urgent conflict. These findings also highlight an important relationship between Resp-1.1 and Resp-1.2: more responsive resolution of a conflict is achieved when the same controller can identify a conflict (Resp-1.1) and decide who resolves it (Resp-1.2).

By following this process for the remaining scenarios in the comparison, additional benefits and tradeoffs can be identified. As an example, Table 42 illustrates these and several other benefits and tradeoffs of architecture options A_4 and A_5 that were identified in this design iteration. The full set of benefits and tradeoffs can be found in Appendix H.

Table 42: Examples of benefits and tradeoffs of A_4 and A_5

ID	**Evaluation Criteria**	**Benefit (+) or Tradeoff (-)**	
		A_4	**A_5**
EC-2.4	**Responsiveness** of trajectory modification decisions when ATM resolves an urgent conflict	⊕	⊖
EC-2.7	**Ease of coordinating** centralization and conflict assignment decisions when switching to centralized decision making	⊕	⊖
EC-2.14	**Ability to maintain alignment** of Controller Assigned to Conflict when deciding who is resolving a conflict	⊕	⊖
EC-2.1	**Responsiveness** of trajectory modification decisions when the aircraft resolve an urgent conflict	⊖	⊕
EC-2.12	**Ability to maintain alignment** of Controller Assigned to Conflict when receiving conflict assignment	⊖	⊕
EC-2.18	**Ability to process** identified conflicts inputs when the workload of the controller processing that input is high	⊖	⊕

Table 42 shows that there are three main benefits of architecture option A_4. First, ATM can make more responsive trajectory modifications when it must resolve an urgent conflict (EC-2.4). As discussed earlier in this section, this is because, when Resp-1.2 is assigned to ATM (as it is in A_4), ATM can immediately begin resolving a conflict if it decides to do so. By contrast, when Resp-1.2 is assigned to the aircraft, ATM may be delayed in resolving the conflict because it needs to wait for the aircraft to request its assistance before it can resolve the conflict.

The second benefit of A_4 is that it is easier to coordinate decisions about when to implement centralized decision making (Resp-1.5) and which controller to assign to conflicts (Resp-1.2) (EC-

2.7). This coordination is necessary because, as shown in the refined conceptual architecture (Figure 39), conflict assignment decisions made in Resp-1.2 depend on the decision made in Resp-1.5 to implement centralized conflict resolution. When centralized decision-making is implemented, then all conflicts should be assigned to ATM. However, when centralized decision-making is not implemented, the prevailing air traffic circumstances should inform a decision about whether ATM or the aircraft should resolve an identified conflict. Thus, coordinating decisions between Resp-1.2 and Resp-1.5 is easier when they are both assigned to ATM (as they are in A_4) because the coordination can occur within ATM. By contrast, when Resp-1.2 is assigned to the aircraft and Resp-1.5 is assigned to ATM (as they are in A_5), this coordination is more difficult because it requires adequate and timely communication between ATM and the aircraft for adequate coordination to occur.

Finally, the third benefit of option A_4 is that it is easier to maintain process model alignment of which controller is assigned to a conflict when deciding who is resolving a conflict (EC-2.14). This is because when Resp-1.2 is assigned to ATM (as it is in A_4), ATM is the sole decision maker for Resp-1.2. By contrast, when Resp-1.2 is assigned to the aircraft, each aircraft maintains its own belief about who they are collectively deciding should resolve a conflict. Although the aircraft are coordinating to make this decision, there are more opportunities in option A_5 for the beliefs of each aircraft about who they are assigning to resolve a conflict to become misaligned.

On the other hand, Table 42 also shows that there are three benefits for architecture option A_5. The first is that the aircraft can make more responsive trajectory modification when they must resolve an urgent conflict (EC-2.1). As discussed earlier in this section, this is because, when Resp-1.2 is assigned to the aircraft (as it is in A_5), the aircraft can immediately begin resolving a conflict if they decide to do so. By contrast, when Resp-1.2 is assigned to ATM, the aircraft may be delayed in resolving the conflict because they must wait for ATM to assign the conflict to them before they can resolve it.

The second benefit of A_5 is that it is easier to maintain process model alignment of which controller is assigned to a conflict when receiving conflict assignments (EC-2.12). This is because when Resp-1.2 is assigned to the aircraft (as it is in A_5), ATM is the only controller that receives conflict assignments from the aircraft. By contrast, when Resp-1.2 is assigned to ATM, the aircraft are the ones receiving assignments from ATM. Because the aircraft each maintain their own process model about what conflicts are assigned to them, there are more opportunities for their process models to become misaligned with respect to which conflicts they need to resolve.

Finally, the last benefit of A_5 is that the aircraft are better able to process inputs from ATM about any identified conflicts when their workload is high (EC-2.18). This benefit is observed because when Resp-1.2 is assigned to the aircraft (as it is in A_5), even if one aircraft is too busy to attend to an input from ATM about an identified conflict, other aircraft can attend to that input. By contrast, when Resp-1.2 is assigned to ATM, ATM is the sole decision maker and must attend to all feedback from any aircraft about an identified conflict.

5.5 Evaluating Support Provided by Framework for Incremental Refinement

Having now completed two iterations of refinement of the ATM architecture for UAM, the goal of this section is to evaluate the ability of this framework to support systems engineers in iteratively refining a system architecture. This comparison therefore evaluates the ability of this framework to address the two other limitations of current approaches for architecture development that were discussed in Section 2.2. First, this framework needs to provide more support to systems engineers than current methods to help them reason about the functions and interactions that need to be included in the system design. Second, this framework also needs to provide better design guidance to support systems engineers in making more informed design decisions.

This framework provides two main types of design support to systems engineers. First, it uses iterative STPA analyses to support incremental learning about how each design decision changes the behavior of the system architecture. Second, the framework provides structured processes for using the insights gained from the STPA analyses to refine the system architecture, enabling more informed downstream design decisions. Figure 44 illustrates how the framework enables these two types of support.

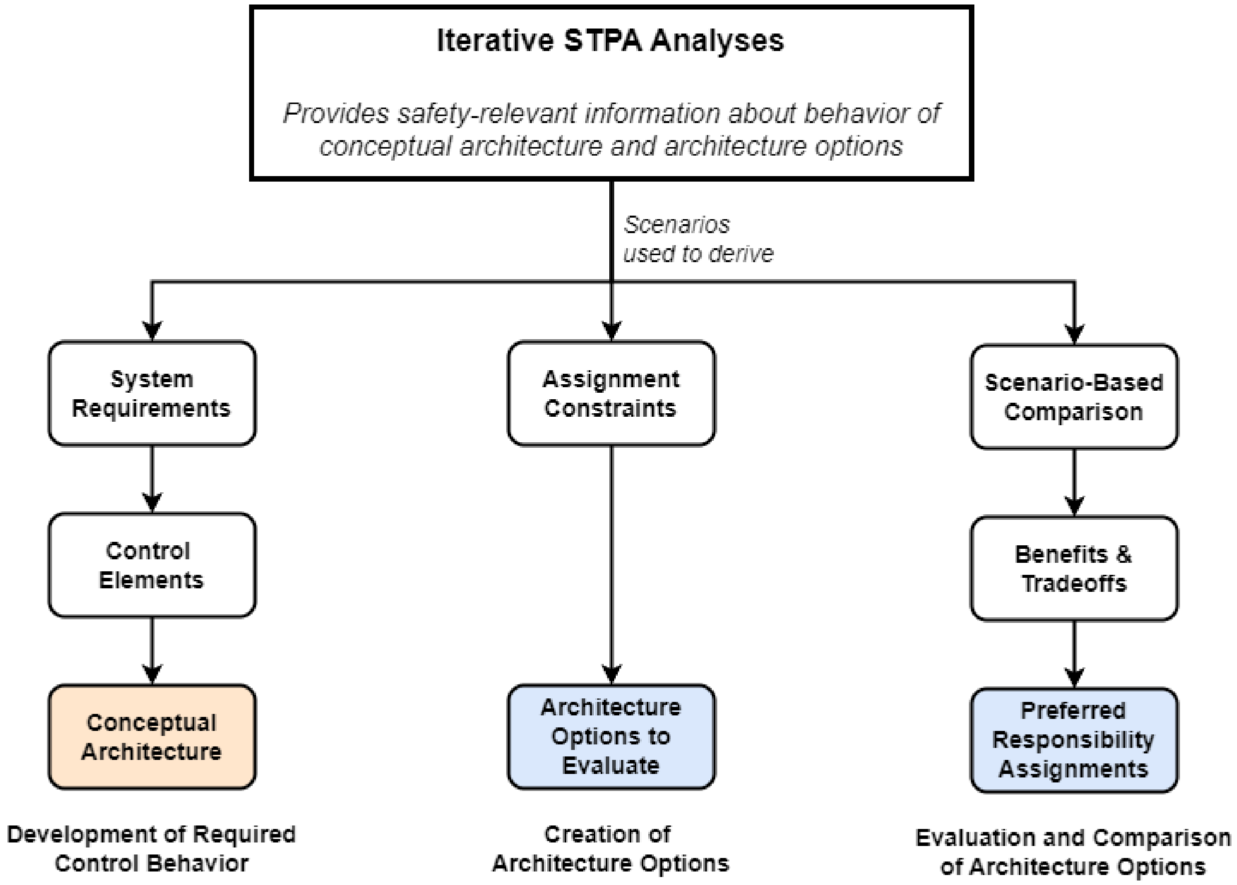

Iterative STPA Analyses
Provides safety-relevant information about behavior of conceptual architecture and architecture options
Scenarios used to derive
System Requirements
Assignment Constraints
Scenario-Based Comparison
Control Elements
Benefits & Tradeoffs
Conceptual Architecture
Architecture Options to Evaluate
Preferred Responsibility Assignments
Development of Required Control Behavior
Creation of Architecture Options
Evaluation and Comparison of Architecture Options

Figure 44: Diagram showing how STPA enables informed architectural design decisions

As shown at the top of Figure 44, STPA provides a structured process for identifying causal scenarios based on the defined losses and hazards for a system. Thus, by using STPA iteratively in this framework to analyze the conceptual architecture as well as each of the architecture options, an analyst or systems engineer can obtain safety-relevant information about the behavior of the conceptual architecture or the architecture options as they design it.

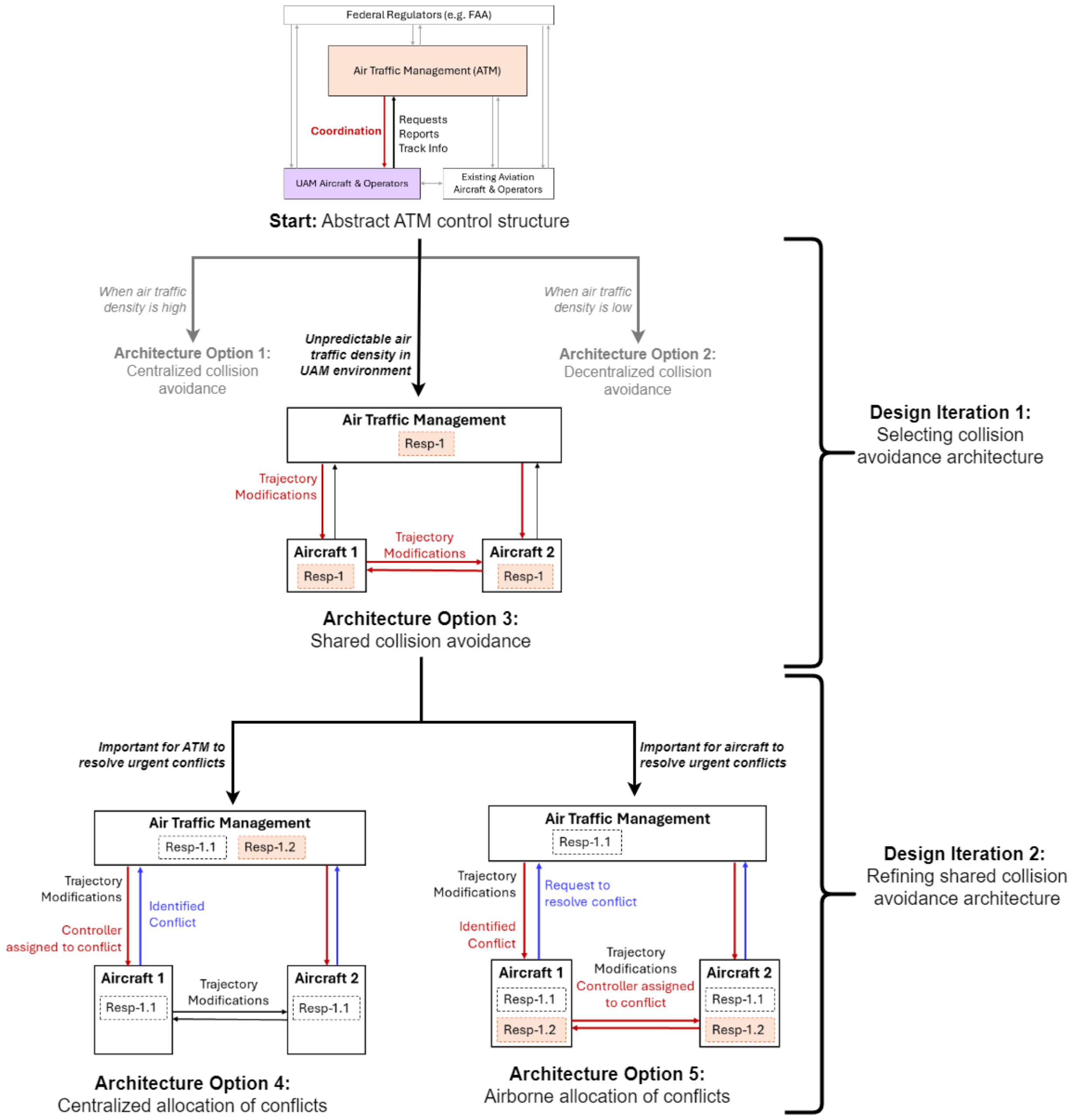
The lower half of Figure 44 then illustrates how the framework helps systems engineers to make more informed design decisions based on what they learn from an STPA analysis. In this architecture development framework, there are three main types of design decisions that are made to create a system architecture, and this framework provides support in making each type of design decision.

The first type of design decisions are the ones that define the required control elements and the relationships between them that are needed to create the conceptual architecture. As shown on the left column of Figure 44, the framework helps systems engineers to make these design decisions by deriving system requirements from the causal scenarios identified by STPA and then deriving the control elements and relationships between them from those system requirements. Only once those control elements and relationships are defined is a conceptual architecture created. This process therefore ensures that every system requirement is informed by a causal scenario that needs to be prevented or mitigated and every element in the conceptual architecture is created to satisfy one or more system requirements. It was this process that informed the creation of a high-level collision avoidance conceptual architecture in Section 4.2 (design iteration 1) and the creation of a refined shared collision avoidance conceptual architecture in Section 5.3 (design iteration 2).

The second type of design decisions are the ones that define what responsibility assignments and therefore which architecture options are worth exploring and evaluating. As shown in the center column of Figure 44, the framework supports the creation of architecture options by deriving assignment constraints from the causal scenarios identified by STPA and then using those assignment constraints to generate architecture options. Thus, the decision to explore each architecture option is informed by the causal scenarios that could be mitigated or prevented by that architecture option. It was this process that informed the creation of architecture options in Section 4.3.1 and Section 4.3.2 (design iteration 1) as well as Section 5.4.1 (design iteration 2).

Finally, the third type of design decisions are the ones made about the preferred way to assign each control responsibility to create a system architecture that best achieves the desired emergent properties. As shown in the right column of Figure 44, the framework supports the selection of a preferred architecture by using the causal scenarios identified by STPA for each architecture option to perform a scenario-based comparison. This comparison helps to identify evaluation criteria and the control-related benefits and tradeoffs of one architecture option compared to another. It is these control-related benefits and tradeoffs that inform decisions about how best to assign the control responsibilities to create the preferred system architecture. Thus, this process ensures that each decision about how best to assign the control responsibilities is informed by the different causal scenarios associated with each architecture option. It was this process that informed the selection of a shared collision avoidance architecture in design iteration 1 based on the benefits and tradeoffs identified in Section 4.3.3. A similar decision could be made for design iteration 2 based on the benefits and tradeoffs identified in Section 5.4.2.

Thus, by applying the framework developed in this dissertation over two design iterations, the results show that this framework can help a systems engineer to start with a very abstract model of a system and incrementally refine and add detail to it while learning about how the architecture's behavior evolves with each design iteration. For the case study used in this research, Figure 45 summarizes how this refinement was done over the two design iterations to create incrementally more refined versions of an ATM architecture for enabling UAM.



Federal Regulators (e.g. FAA)
Air Traffic Management (ATM)
Coordination
Requests
Reports
Track Info
UAM Aircraft & Operators
Existing Aviation Aircraft & Operators
Start: Abstract ATM control structure
When air traffic density is high
Architecture Option 1: Centralized collision avoidance
Unpredictable air traffic density in UAM environment
When air traffic density is low
Architecture Option 2: Decentralized collision avoidance
Design Iteration 1: Selecting collision avoidance architecture
Air Traffic Management
Resp-1
Trajectory Modifications
Aircraft 1
Aircraft 2
Architecture Option 3: Shared collision avoidance
Important for ATM to resolve urgent conflicts
Important for aircraft to resolve urgent conflicts
Design Iteration 2: Refining shared collision avoidance architecture
Resp-1.1
Resp-1.2
Identified Conflict
Controller assigned to conflict
Request to resolve conflict
Architecture Option 4: Centralized allocation of conflicts
Architecture Option 5: Airborne allocation of conflicts

Figure 45: Iterative refinement of the ATM architecture across design iterations

The first design iteration (presented in Chapter 4) started with an abstract ATM control structure and applied the framework to determine that a shared collision avoidance ATM architecture would be preferable for UAM. This shared collision avoidance ATM architecture therefore refined the initial abstract ATM control structure by defining more specific control responsibilities and more detailed control actions and feedback that will be needed to safely manage UAM air traffic.

Then, the second design iteration (presented in Chapter 5) started with the shared collision avoidance ATM architecture developed in iteration 1 and further refined it by identifying two possible options for implementing Resp-1.2 (the responsibility for assigning conflicts to controllers to resolve). Either of these options is a further refinement of the shared collision avoidance ATM architecture chosen in iteration 1 because they define a more detailed set of control responsibilities that are needed to enable shared collision avoidance. In addition, they define more specific control actions and feedback that need to be exchanged between ATM and the aircraft as well as between the aircraft to enable adequate coordination between them.

These two design iterations therefore demonstrate that the structured processes provided by this framework provide support to help systems engineers refine a system architecture and learn about the system architecture they are developing as they create it.

In addition to being able to refine the system architecture incrementally over time, an additional capability offered by this framework is the ability to reconsider and revise past decisions. This capability is achieved because of the incremental way that a system architecture is refined when applying this framework and the traceability that is maintained during each step.

For example, recall that the purpose of performing design iteration 2 was to confirm the feasibility of implementing a shared collision avoidance architecture and to learn more about the potential tradeoffs of a shared collision avoidance architecture. When architecture options 4 and 5 (created in design iteration 2) are analyzed and compared, a systems engineer could make one of two choices based on those comparison results. One choice they could make is that the benefits of implementing a shared collision avoidance architecture are worth the tradeoffs that have been identified and they can continue to refine the ATM architecture as was done in this research.

However, a systems engineer could also decide that the comparison results show that a shared collision avoidance architecture introduces too many risks or implementation challenges. As a result, contrary to what was initially believed, they might decide that the benefits are not worth the tradeoffs or the effort required to address the risks. If such a decision is made, the traceability that is maintained when applying this framework would allow a systems engineer to return to the comparison performed in iteration 1, re-examine the comparison results, and decide that a different architecture option may be preferable instead. This revision of a past decision is illustrated in Figure 46.

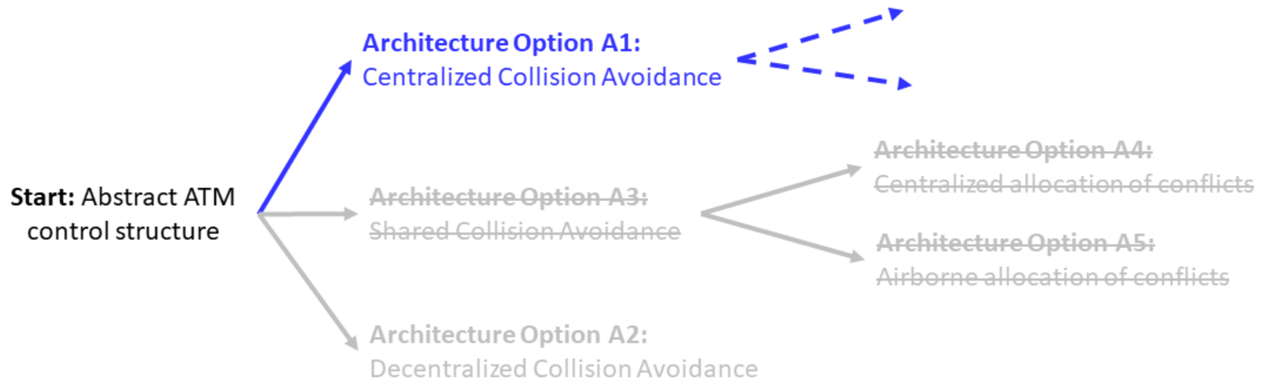

Architecture Option A1:
Centralized Collision Avoidance
Start: Abstract ATM control structure
~~Architecture Option A3:~~
~~Shared Collision Avoidance~~
~~Architecture Option A4:~~
~~Centralized allocation of conflicts~~
~~Architecture Option A5:~~
~~Airborne allocation of conflicts~~
Architecture Option A2:
Decentralized Collision Avoidance

Figure 46: Revising a past architecture option selection

As shown in Figure 46, although an initial decision was made to select architecture option A_3 in design iteration 1, a systems engineer could return to that decision at any point and re-evaluate their architecture preferences based on what was learned about the shared collision avoidance architecture. A revised decision can then be made. For example, architecture option A_1 might be considered preferable now in light of what was learned about the challenges and risks involved in implementing a shared collision avoidance architecture. Thus, the traceability provided by this framework enables revisions and reconsiderations such as this one.

5.6 Summary

This chapter described the results of the second design iteration performed in this research. The goal of this second design iteration was to refine the high-level shared collision avoidance architecture that was selected in design iteration 1 to better define how ATM and the aircraft will share responsibility for collision avoidance and work together to adequately resolve conflicts. Although this architecture represents a more collaborative approach to shared collision avoidance than what exists in today's ATC system, the Uberlingen mid-air collision in 2002 showed what could go wrong when shared responsibility for collision avoidance is not adequately controlled in the system architecture.

This second design iteration therefore began with an analysis of the high-level shared collision avoidance architecture using STPA-Teaming to analyze how ATM and the aircraft might collectively be unable to adequately resolve conflicts. Based on the results from this analysis, additional system requirements were generated that described the additional safety constraints necessary to ensure adequate control over the shared responsibility for collision avoidance. These additional requirements were then used in conjunction with the requirements identified during iteration 1 to create a refined conceptual architecture that defined the control behavior needed to prevent unsafe collective decision making by ATM and the aircraft when resolving conflicts.

Based on the refined conceptual architecture, two architecture options for implementing Resp-1.2 were compared. Architecture option A_4 represented a ground-based conflict assignment architecture where ATM decides whether it or the aircraft should resolve an identified conflict. Architecture option A_5 represented an airborne conflict assignment architecture where the aircraft decide whether they want to resolve a conflict themselves or request the assistance of ATM to resolve it. By comparing these two architecture options, a series of benefits and tradeoffs of each option were identified.

Finally, the full set of results from both design iterations were used to evaluate whether the framework provides better support to systems engineers in reasoning about what needs to be included in their system design and to help them make more informed decisions. The results from both design iterations showed that the use of iterative STPA analyses and the structured processes provided by this framework allows systems engineers to incrementally learn about the architecture they are creating and make more informed design decisions as they refine the system architecture. In addition, because the rationale for each design decision is maintained at each step of the framework, it is easier for systems engineers to revisit and revise past design decisions. These findings therefore provide support for hypothesis 2 of this dissertation and demonstrate that the framework can be used to iteratively develop and refine the architecture for a system.

Hypothesis 2: A systems-theoretic approach can support making informed design decisions to iteratively develop and refine the architecture for a system

In both design iterations conducted in this research, there are numerous assumptions that underlie the design decisions that were made and the comparison results that were generated. While some of these assumptions were highlighted in this chapter and the previous one, little was done to account for them and ensure they are not violated as architecture development progresses. In the next chapter, a supporting framework for identifying and ensuring the validity of underlying assumptions that was developed in this research will be described and demonstrated.

Chapter 6 Ensuring the Validity of Underlying Assumptions

In the prior chapters of this dissertation, the importance of identifying underlying assumptions and ensuring they remain valid over time has been emphasized. However, no guidance was provided as to how to identify assumptions or what to do with the identified assumptions to ensure they remain valid. To address these challenges, this research developed a supporting framework for identifying underlying assumptions and accounting for them during architecture development to help systems engineers avoid flaws arising because of assumptions becoming invalid.

This chapter develops and demonstrates this supporting assumptions framework and is organized as follows. Section 6.1 provides an overview of the role of assumptions in architecture development. Section 6.2 describes the guiding prompts that were developed to provide systems engineers with guidance for identifying assumptions at each step of architecture development. Section 6.3 then demonstrates how these guiding prompts can be applied to identify assumptions underlying the various design decision made to create and refine the ATM architecture for the UAM case study presented in Chapter 4 and Chapter 5. These examples illustrate the different types of assumptions that can be identified. Finally, Section 6.4 describes how to ensure that the identified assumptions are accounted for and monitored over time as architecture development proceeds.

6.1 The Role of Assumptions in Architecture Development

There are several reasons why we make assumptions during system design. Sometimes, assumptions record important information that explains the reasoning behind a design decision [110]. For example, when deciding whether a design decision will be adequate in preventing unsafe or undesirable behavior, analysts or systems engineers might make assumptions to justify that decision. Alternatively, assumptions can also be used to capture a designer's expectation or prediction about what might happen in the future [111]. This includes assumptions about what the system's operating environment might be once the system is fielded or how that environment might evolve over time. It also includes assumptions about how the system might interact with or impact its operating environment once it is operational.

Regardless of the reason they are made, assumptions play an important role in design, especially during the early stages, because they allow a systems engineer to make design decisions despite the uncertainties they might face during the development process. These uncertainties might occur because some aspects of the system design have not yet been determined or the system's impact on its environment is not yet known with certainty. In addition, because designing a complex system can be considered to be a "wicked" planning problem [112, 113], a systems engineer may not know all the factors that will be important to consider about the design until they start the design process itself.

Although assumptions help to mitigate these sources of uncertainty, it is important to recognize that the design decisions made using these assumptions become contingent on those assumptions remaining valid [16]. This is because the design decisions will only have their intended effect if the assumption remains valid. Once an assumption becomes invalid, any design decisions made based on it may now be flawed.

The NTSB emphasizes this important aspect of assumptions in a safety recommendation report they published for the 737 MAX accidents in 2019 [114]. In the report, the NTSB found that Boeing made flawed assumptions about the behavior of the Maneuvering Characteristics Augmentation System (MCAS) and the ability of the flight crew to respond in the event of unintended MCAS activation. As a result of the flawed assumptions Boeing made, flight crews were unable to recover control of the aircraft as Boeing had assumed when unintended MCAS activations actually occurred [114]. Thus, the 737 MAX accidents illustrate the importance of being able to make explicit the assumptions underlying the system design and ensure that their validity can be verified and monitored over time [16, 111].

Despite the importance of these underlying assumptions, current methods for architecture development provide minimal, if any, guidance on how to identify these underlying assumptions or how to account for them during architecture development. To address these limitations, better guidance must first be provided to help systems engineers identify assumptions underlying their architectural design decisions.

6.2 A Framework for Identifying Underlying Assumptions

To determine how to provide such guidance, a method called Assumptions-Based Planning (ABP) [34] serves as a useful reference. ABP was created to identify the assumptions underlying a business or organizational plan and develop a plan for ensuring the validity of those assumptions over time [111]. In ABP's taxonomy, assumptions can be classified into two categories: (1) assumptions about problems and (2) assumptions about solutions [111]. The first category consists of assumptions an organization makes about the problems it believes it will encounter and these typically describe the environment in which it expects to be conducting its business. By contrast, the second category consists of assumptions an organization makes about how it will address those problems (i.e., the solutions).

These two categories of assumptions can be extended to engineered systems as well. Instead of making assumptions about "problems" and "solutions", systems engineers make assumptions about (1) the *environment* that the future system will operate in and (2) how the *system* itself will solve a problem or meet a set of needs. For example, the creation of UAM to provide transportation services in urban areas assumes that UAM provides a useful mode of transportation for an urban community (a system assumption) and that the public will need and be willing to use UAM services (an environmental assumption). Thus, when identifying assumptions during architecture development, it can be helpful to consider if either of these types of assumptions are being made.

Another key aspect of ABP is the methods for identifying assumptions. [111] suggests three methods: (1) telling actions the long way, (2) strategic assumption surfacing and testing, and (3) discovery-driven planning. Although these methods use different guidewords and processes, the fundamental strategy for helping an analyst identify assumptions is the same: rationalize the action or design decision and determine what needs to be true for the decision to have the desired effect or outcome.

This research applies this strategy to create a set of guiding prompts to help systems engineers consider any assumptions they might be making about the system or environment that need to remain valid for the system to behave as intended. These guiding prompts are shown in Table 43.

Consistent with the strategy used by ABP, the guidance in Table 43 prompts systems engineers to consider what needs to be true or what needs to exist for the design decision to have the desired effect. These are the assumptions that, if invalid, might compromise the ability of the system to adequately enforce the safety constraints or achieve adequate control over the behavior of the system or its components. Because each step of architecture development involves different types of design decisions, these guiding prompts are tailored to the various steps in the architecture development framework described in Chapter 3.

Table 43: Guiding prompts for identifying underlying assumptions

Architecture Development Step	**Guidance for Identifying Assumptions (Either system or environment assumptions)**
Performing STPA Analyses	• What assumptions is the system boundary based on? • What assumptions is the STPA control structure based on? • What assumptions are being made when generating the UCAs and scenarios?
Identifying System Requirements	• What assumptions must be true for the requirements to be effective at preventing undesirable behavior?
Developing the Conceptual Architecture	• What is being assumed about the behavior of a responsibility for it to meet the system requirements? • What assumptions must be true for the process model parts to be maintained and kept updated? • What is assumed to be available to receive as feedback from the environment? • What is being assumed for the defined control actions to be effective?
Exploring and Comparing Architecture Options	• What must be true for the architecture to implement the system-level behavior? • What is being assumed about how the architecture will be implemented during detailed system design? • What is being assumed to decide if an architecture will resolve a given scenario?

6.3 Using the Framework for Identifying Underlying ATM Assumptions

To illustrate how the guidance shown in Table 43 should be used during architecture development, this section presents examples of assumptions identified at each step of the architecture development process. Each identified assumption is labeled in square braces as either an environmental or system assumption. These labels illustrate that throughout the architecture development process, both types of assumptions can be identified.

6.3.1 Assumptions Underlying Initial STPA Analysis

When performing an STPA analysis, underlying assumptions primarily serve to justify the elements of the control structure as well as rationalize the identification of UCAs and scenarios. In this section, examples of assumptions made during an STPA analysis are shown for the initial STPA analysis of the NAS that was described in Chapter 4.

The first place in the initial STPA analysis where assumptions are made is when creating the abstract NAS control structure (Figure 24). Because this future NAS does not yet exist, these assumptions represent expectations about how ATM and regulators would interact with both existing aviation and UAM air traffic and inform the control actions and feedback that are included in the control structure. Table 44 shows examples of modeling decisions made to create the NAS control structure shown in Figure 24 and their underlying assumptions.

Table 44: Example assumptions underlying the NAS control structure

Modeling Decision	**Underlying Assumption**
Control actions and feedback between existing aviation operations and ATM and regulators are modeled to be similar to those today	It is assumed that aviation will follow ATC and FAA rules that are fundamentally similar to those used today (e.g., ADS-B equipage requirements) [Environment assumption]
Sharing of position/ID/speed data between aircraft	It is assumed that, at a minimum, ADS-B-like data will need to be shared between aircraft to ensure that UAM aircraft can safely interact with existing aviation operations [Environment assumption]
Control actions and feedback between ATM and UAM operations	It is assumed that UAM air traffic will need to operate on or near conventional airports. ATM will therefore need some control over UAM aircraft to manage their arrival/departure alongside existing aviation operations [System assumption]
Control actions and feedback between regulators and UAM operations	It is assumed that, like commercial air carriers today, UAM operations and aircraft will be certified by regulators [System assumption]

Once the control structure was created, assumptions were also made to identify UCAs and scenarios. Like when creating the control structure, these UCAs and scenarios describe unsafe behavior that is expected to occur in the NAS that would exist in the future with UAM integrated into it. Thus, assumptions were made to inform the various contexts and operating conditions contained in UCAs in which UAM air traffic might be managed. Similarly, assumptions also informed the different types of interactions that UAM air traffic might have with each other or with other air traffic. Table 45 shows examples of UCAs and scenarios identified in the initial STPA analysis and the underlying assumptions used to generate them.

Table 45: Example assumptions underlying UCAs and scenarios identified in initial STPA

Unsafe Control Action (UCA) or Causal Scenario (CS)	Underlying Assumption
UCA-1.5: Air Traffic Management does not coordinate the movements of UAM aircraft when they are about to fly into a section of airspace where air traffic must be excluded (e.g., for safety or security reasons)	UAM traffic flow will need to be changed or restricted at times to meet demands imposed by the environment. These include but are not limited to: clearing path of first responders to emergencies, and Temporary Flight Restrictions (TFR) for major public events, VIP protection, and other circumstances [Environment assumption]
UCA-1.13: Air Traffic Management does not coordinate UAM aircraft when inclement weather is approaching that could interfere with flight operations	UAM operations will experience the span of year-round weather conditions, visibility conditions as well as both day and night operations [Environment assumption]
CS-1.8.1-4: Air Traffic Management does not realize that another NAS user has a time-critical mission to execute (e.g., emergency medical flight) and believes that the other NAS user's mission can be delayed for the UAM aircraft and therefore wrongly decides not to provide coordination to avoid the delay for the other NAS user	UAM aircraft will operate in urban areas where public safety and other missions may also be operating at similar altitudes as UAM flights [Environment assumption]
CS-1.8.4-1: Air Traffic Management is notified of a UAM flight shortly before its departure time and there is not enough time for it to issue adequate coordination before the departure time. As a result, a UAM aircraft interferes with the operations of another airspace user.	Although some UAM air traffic may be predictable (e.g., regularly scheduled shuttle flights), many UAM flights will occur on an on-demand basis with little regularity [Environment assumption]

6.3.2 Assumptions Underlying System Requirements and Conceptual Architecture

When identifying system requirements, underlying assumptions describe what must be true for the requirements to have the desired effect of mitigating or preventing the identified UCAs or scenarios. Sometimes, these assumptions describe certain aspects of a system's environment that are needed for a requirement to be implementable. Other times, these assumptions describe certain aspects of a system's behavior that are needed for a requirement to be sufficient at preventing an STPA scenario. Table 46 shows some examples of system requirements identified in Chapter 4 and their underlying system and environmental assumptions.

Table 46: Examples of system requirements and their underlying assumptions

System Requirement	**Underlying Assumptions**
Req-3: ATM system shall ensure that sufficient capacity is available to detect and coordinate all aircraft that have or will need access to the airspace	• Assumes that surges in demand for flights will occur with at least <TBD mins> of advance notice for the NAS to implement plans to mitigate system impacts [Environment Assumption] • Assumes that this requirement is carried out in conjunction with the airspace access management requirement, especially whenever demand nears capacity limits [System Assumption]
Req-4: ATM system shall coordinate the movement of aircraft to resolve any potential conflicts	• Assumes that UAM flights are known within <TBD> time of desired departure [Environment Assumption] • Assumes coordination decision can be made within <TBD> time [System Assumption]
Req-8: ATM system shall only allow as many users to access the airspace as it is capable of detecting, tracking and coordinating	• Assumes that this requirement is used in conjunction with capacity management to ensure that all aircraft can be adequately coordinated (e.g., during surge times) [System Assumption]
Req-10: ATM system shall account for intended movements of aircraft in addition to current trajectories to detect potential collisions	• Assumes that aircraft are willing to share their intended trajectories for at least <TBD time> into the future (e.g., no privacy concerns) [Environment Assumption]

Similarly, when developing the conceptual architecture, underlying assumptions describe what must be true for the various control responsibilities and their associated control actions and feedback in the conceptual architecture to meet the system requirements. Figure 47 and Figure 48 show example assumptions identified for two different responsibilities and their associated control actions and feedback.

Responsibility Resp-1: Coordinate the movement of aircraft to prevent undesirable interactions

Summary of Desired Behavior: A conflict is defined as a situation where two aircraft will pass within <TBD> distance horizontally or vertically of each other.

If a potential collision is detected, aircraft should be provided with new/updated trajectories that minimize delays or mission impact while resolving the potential collision/interference. Coordination should account for aircraft capabilities, mission constraints, delays in executing coordination as well as the future movements of/coordination provided to other aircraft.

If an aircraft is non-communicative, use its last communicated trajectory to coordinate the movements of other aircraft to prevent collision

> *System Assumption:* Assumes that an aircraft experiencing an emergency will be granted highest priority access to the airspace they need to address the emergency
>
> *System Assumption:* Assumes that access priorities are considered when coordinating aircraft movements to prevent collisions and when managing airspace access
>
> *System Assumption: Assumes* that coordination decisions can be made within <TBD> time

Control Actions: Trajectory modifications

> *Environmental Assumption:* Assumes that UAM will be best served by moving toward trajectory-based operations (TBO)

Feedback: Basic aircraft telemetry, aircraft type and capabilities, future trajectory

> *Environmental Assumption:* ADS-B-like tracking data will be available on all aircraft in UAM airspace

Figure 47: Example 1 of assumptions underlying the definition of Resp-1

Responsibility Resp-4: Only allow as many users to access the airspace as it is capable of detecting, tracking and coordinating

Summary of Desired Behavior: Approve/decline requests for access to airspace based on how many active flights have already been approved and the established airspace access priorities for different users/missions. Requests should be managed to avoid exceeding the maximum capacity of the NAS to track and coordinate them as well as ensure that sufficient space is available for alternative movement options for each aircraft. This includes both immediate approval/denial of access as well as longer-term access planning

> *System Assumption:* Assumes this responsibility is coordinated with the responsibility for determining alternative movement options to ensure that access management considers the space needed for alternative movement options
>
> *Environmental Assumption:* Assumes at least some flights are known with at least <TBD> advanced notice so that there is time to negotiate changes to the flight plan

Control Actions: Approve/decline access request, Flight plan modification options

> *Environmental Assumption:* Assumes that UAM flights will have some amount of operational flexibility to allow ATM to propose flight plan modifications such as departure time changes or flight route changes that are acceptable

Feedback: Flight plan, mission and operational constraints, possible movement options, capacity of ATM, Congestion level

> *Environmental Assumption:* Assumes that UAM aircraft will be willing to share flight plans and mission and operational constraints to enable this feedback

Figure 48: Example 2 of assumptions underlying the definition of Resp-4

6.3.3 Assumptions Underlying Comparison of Architecture Options

Finally, when exploring and comparing architecture options, one of the main steps where assumptions play a critical role is in the comparison of architecture options. As described in Chapter 3, comparing architecture options involves deciding whether a given STPA scenario occurs for each architecture option. However, at this stage in the architecture development process, some design details needed to inform this decision may not have been made yet. Thus, assumptions may need to be made about the environment or the behavior of the system to decide if a scenario is mitigated or prevented by that architecture option.

As a result, however, the ability of that architecture option to prevent each scenario becomes contingent on those assumptions remaining valid. Especially if one of these architecture options is chosen for further development, it is extremely important to ensure that any downstream design decisions do not violate the assumptions associated with that architecture option. This is because if an assumption does become violated, then the architecture might have a flaw or unsafe behavior that was assumed not to exist.

In Section 4.3.3, Table 21 showed examples of assumptions made when comparing the centralized and decentralized collision avoidance architecture options to decide whether a scenario was resolved by an architecture option. Table 47 replicates the assumptions shown in Table 21 and labels each assumption as either a system assumption or an environmental assumption to show that both system and environmental assumptions can be identified.

Table 47: Examples of assumptions underlying comparison decisions

ID	Assumption	Assumption Type
A-1	It is assumed that ATM will not have to coordinate conflicts as frequently because it has broader situational awareness of the future state of the airspace and can better anticipate and resolve multiple conflicts in a more coordinated fashion.	System Assumption
A-2	It is assumed that UAM aircraft would have onboard sensing capable of detecting ground hazards with enough range to allow time for the aircraft to respond to avoid a collision with the ground hazard	Environmental Assumption
A-3	It is assumed that with the aircraft sharing responsibility for preventing conflicts with ATM, a component failure (e.g., on ATM or on one of the aircraft) should not compromise the ability of other aircraft to prevent conflicts	System Assumption
A-4	It is assumed that even if an initial set of aircraft are preoccupied with resolving a set of conflicts, any new aircraft would identify the conflict and coordinate its own set of trajectory modifications to avoid the other group of aircraft	System Assumption

6.4 Deriving Requirements from Underlying Assumptions

Having identified the underlying assumptions, a process is now needed to ensure that these assumptions are not violated at any point during the system's lifecycle and the methods available for doing so depend on the type of assumption being made as shown in Figure 49.

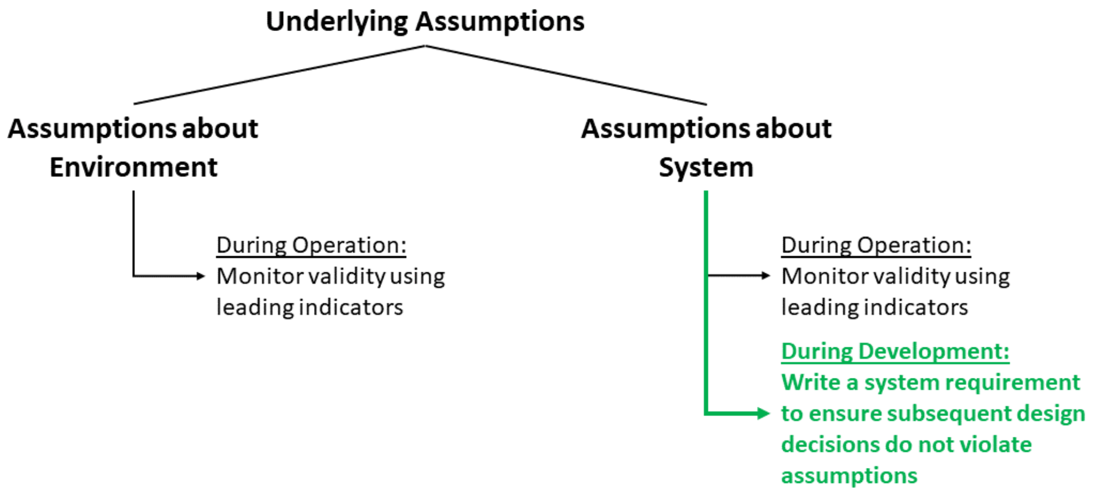

Underlying Assumptions
Assumptions about Environment
Assumptions about System
During Operation:
Monitor validity using leading indicators
During Operation:
Monitor validity using leading indicators
During Development:
Write a system requirement to ensure subsequent design decisions do not violate assumptions

Figure 49: Methods for ensuring the validity of underlying assumptions

For both assumptions about the environment and assumptions about the system, they should be monitored during operations using assumptions-based leading indicators [16]. These indicators essentially serve as warning signs indicating when an underlying assumption may be close to or already violated so that decision makers can take corrective action to prevent an actual accident or loss from occurring.

For assumptions about the system, however, more can be done besides just monitoring them during operations. Their validity can also be enforced during development because the system's design is under the control of the systems engineer. Thus, a requirement can be derived from each underlying system assumption that describes a constraint that will prevent the assumption from being violated. Table 48 shows examples of system assumptions and the system requirements derived from them.

These additional derived requirements can then be used in two ways to ensure they remain valid as system development progresses. First, the architecture development framework can be applied to these derived requirements so that the conceptual architecture and system architecture are modified to account for them. Figure 50 illustrates how additional control elements are derived from the derived requirements.

Second, during verification and validation (V&V) of the system after design is complete, these additional derived requirements can be verified to check the validity of the underlying assumptions. If the additional derived requirements are all verified successfully, that would imply the underlying assumptions have not been violated.

Table 48: Examples of system assumptions and derived requirements

System Assumption	**Derived Requirement**
Assumes that access priorities are considered when coordinating aircraft movements to prevent collisions and mitigate congestion as well as when managing airspace access	**Req-20:** ATM system shall consider access priorities when selecting trajectory modifications or managing access to the airspace
Assumes that trajectory modification decisions can be made within <TBD> time	**Req-49:** ATM system shall be able to make trajectory modification decisions within <TBD> time
Assumes that if selected trajectory modifications are found to not be adequate, those trajectory modifications are evaluated again to ensure that collision risks are adequately mitigated	**Req-50:** ATM system shall ensure that if selected trajectory modifications are found to not be adequate, those trajectory modifications are evaluated again to ensure that collision risks are adequately mitigated
Assumes there is enough space available to keep other aircraft away from a non-communicative aircraft	**Req-69:** ATM system shall ensure that there is enough spare airspace available to keep other aircraft away from a non-communicative aircraft
Assumes that if trajectory modifications are not acknowledged within <TBD> time, the	**Req-93:** ATM system shall re-evaluate trajectory modification(s) associated with a

conflict associated with that modification will be flagged for re-evaluation	conflict if the trajectory modification(s) are not acknowledged within <TBD> time

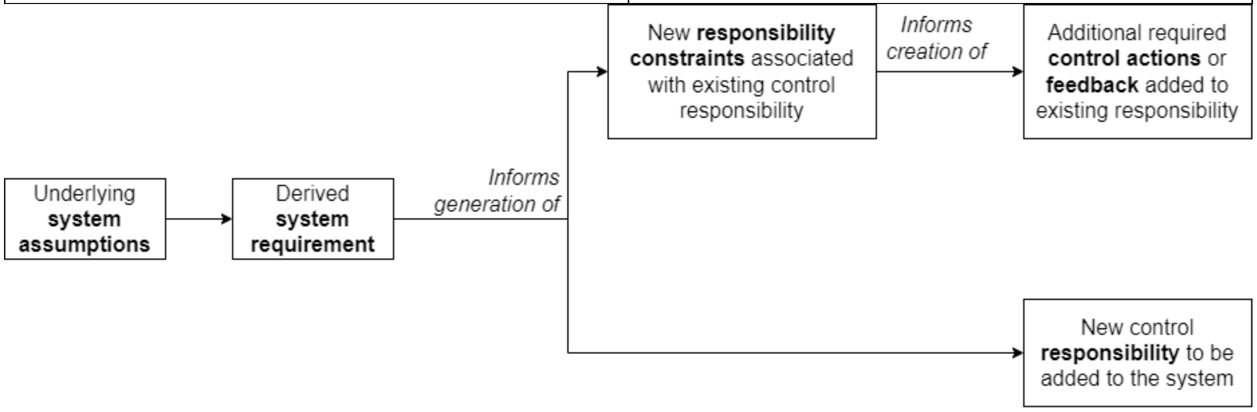
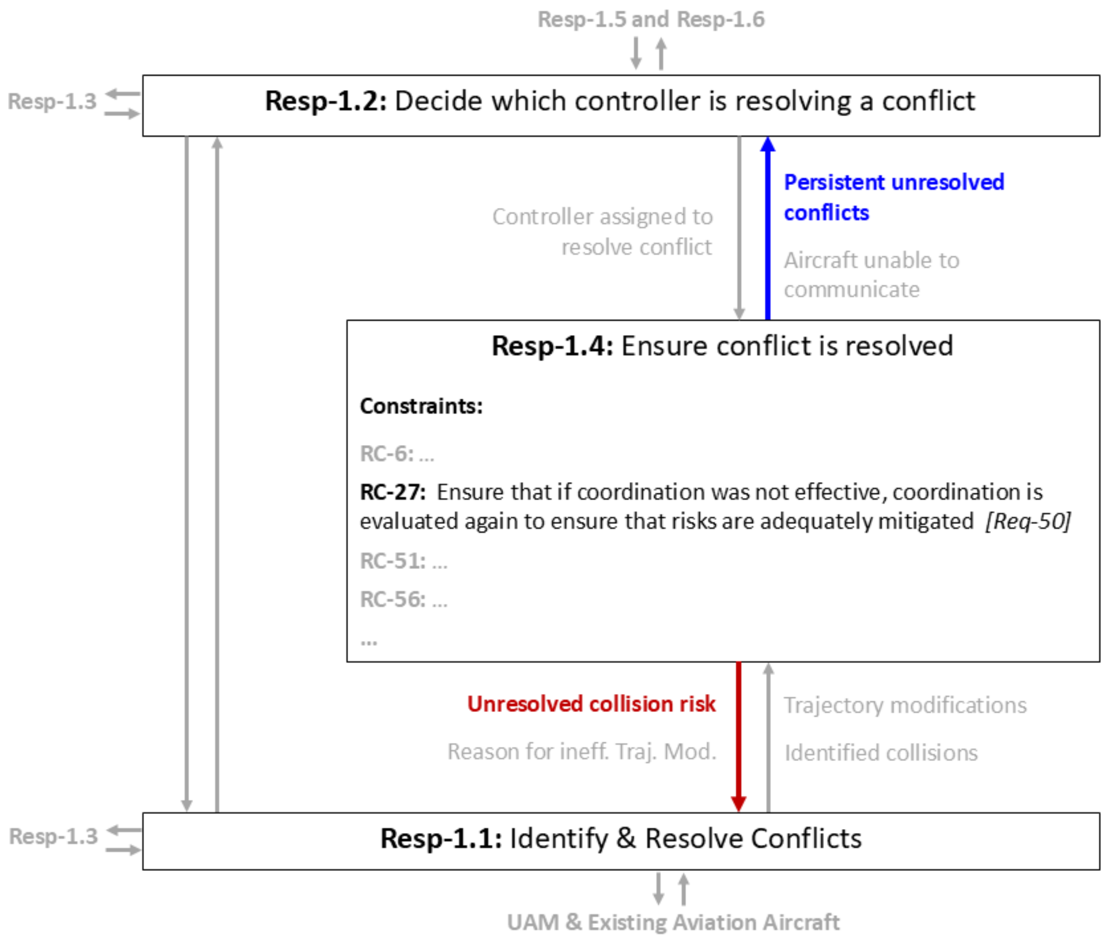

Underlying **system assumptions**
Derived **system requirement**
Informs generation of
New **responsibility constraints** associated with existing control responsibility
Informs creation of
Additional required **control actions** or **feedback** added to existing responsibility
New control **responsibility** to be added to the system

Figure 50: Accounting for underlying assumptions during development

As shown in Figure 50, once system requirements have been derived from the underlying system assumptions, they can then be used to generate either (1) new responsibility constraints associated with an existing control responsibility or (2) new control responsibilities. New control actions and feedback or new responsibilities can then be added to the conceptual architecture to account for these new responsibilities or responsibility constraints.

To illustrate how this is done, consider the derived requirement Req-50 in Table 48. This requirement was generated in design iteration 1 and was derived from an assumption that if trajectory modifications are found to be inadequate for preventing a conflict, those trajectory modifications will be re-evaluated to identify revised trajectory modifications for preventing that conflict. It is therefore important that this requirement is met by the conceptual architecture to ensure its associated underlying assumption remains valid as architecture development proceeds.

Although Req-50 was generated in design iteration 1, it was used to inform the inclusion of some control elements in the refined conceptual architecture in design iteration 2. Figure 51 illustrates part of the conceptual architecture created in design iteration 2 (originally shown in Figure 39) and highlights the control elements that were included to meet Req-50.



Resp-1.5 and Resp-1.6
Resp-1.3
Resp-1.2: Decide which controller is resolving a conflict
Controller assigned to resolve conflict
Persistent unresolved conflicts
Aircraft unable to communicate
Resp-1.4: Ensure conflict is resolved
Constraints:
RC-6: …
RC-27: Ensure that if coordination was not effective, coordination is evaluated again to ensure that risks are adequately mitigated [Req-50]
RC-51: …
RC-56: …
…
Unresolved collision risk
Reason for ineff. Traj. Mod.
Trajectory modifications
Identified collisions
Resp-1.3
Resp-1.1: Identify & Resolve Conflicts
UAM & Existing Aviation Aircraft

Figure 51: Accounting for Req-50 in design iteration 2 conceptual architecture

Req-50 describes what needs to happen if a conflict is found to have not been adequately resolved. Thus, Req-50 is categorized as a constraint requirement (not a control requirement) and the responsibility constraint RC-27 is generated and associated with Resp-1.4 (the responsibility for ensuring that conflicts are adequately resolved).

Then, to ensure that Resp-1.4 meets the constraint RC-27, two control elements were added: (1) a control action called *Unresolved collision risk* was added between Resp-1.4 and Resp-1.1 and (2) a piece of feedback called *Persistent unresolved conflicts* is added between Resp-1.4 and Resp-1.2. Together, these two control elements allow Resp-1.4 to notify or prompt the relevant responsibilities to reconsider their control decisions to ensure that some action is taken to adequately resolve the conflict. The control action *Unresolved collision risk* ensures that the controller assigned to resolve that conflict (i.e., either ATM or the aircraft) are notified to revise their selected trajectory modifications if the one they chose originally did not adequately resolve the conflict. Similarly, the feedback *Persistent unresolved conflicts* ensures that if a conflict remains unresolved after several attempts to select revised trajectory modifications, Resp-1.2 can be prompted to re-evaluate the controller that it assigned to the conflict to determine if an alternative controller would be better equipped to resolve the conflict.

Although the example shown in Figure 51 is relatively simple, it illustrates how underlying system assumptions made in an earlier iteration of design can be accounted for in the conceptual architecture and appropriate control elements added in later iterations to ensure those assumptions remain valid as the system architecture is refined.

6.5 Summary

This chapter developed and demonstrated a supporting framework to help systems engineers more easily identify the assumptions they are making and account for them during the architecture development process. Recognizing that identifying assumptions is a challenging task, this supporting framework provides specific guidance for identifying assumptions that is tailored to the different steps in the safety-driven architecture development framework developed in this research. By using this framework, assumptions about both the system and environment can be identified when performing STPA analyses, identifying system requirements, developing the conceptual architecture, and exploring and comparing architecture options.

Once assumptions are identified, one way to ensure their validity over time is to generate assumptions-based leading indicators to monitor them during system operation. Assumptions about the system can also be accounted for during architecture development by deriving additional system requirements from these underlying system assumptions. These additional derived requirements can then be used to inform the addition of control elements to the conceptual and system architecture. In addition, the validity of these assumptions can be more easily checked during system verification and validation when the derived requirements are verified.

Chapter 7 Conclusions & Future Work

Developing complex systems today is becoming more challenging than ever before. Not only is greater functionality and productivity being demanded from these systems, but there is also an increasing desire to use automation and software to enhance their capabilities. As a result, systems have gotten more complex, interconnected, and reliant on software while being increasingly expected to exhibit emergent properties such as safety and security. Unfortunately, designing these systems to exhibit these properties is challenging because existing methods do not provide the necessary design support to help systems engineers design these properties into their system architectures.

The objective of this dissertation was to address this problem by creating an architecture development framework that provides structured and systematic processes for creating and assessing system architectures. Unlike current methods, this architecture development framework was developed based on Systems Theory and provides appropriate types of support to help systems engineers incrementally develop and refine a system architecture for their system.

The key idea underlying this new framework is that a system should be designed to prevent unsafe or undesirable behavior. Thus, the first part of the framework involves performing an initial STPA that identifies preliminary scenarios describing how unsafe behavior could occur. Then, the behavioral design process provides a structured way to use these STPA scenarios to derive system requirements and a conceptual architecture that describes the control behavior needed to prevent unsafe behavior from occurring. Finally, the structural design process provides a systematic method for exploring and comparing possible architecture options to implement the conceptual architecture. By comparing these architecture options, control-related benefits and tradeoffs between the architecture options are identified. Ultimately, these benefits and tradeoffs can be used by systems engineers to inform their decisions about how to architect the system to best achieve the desired emergent properties such as safety. When this framework is applied iteratively, the system architecture can be incrementally refined in a top-down manner and safety can be designed into it from the beginning.

This architecture development framework was applied over two design iterations to develop an ATM architecture for the NAS that can manage UAM air traffic alongside existing air traffic. The first design iteration focused on developing a high-level collision avoidance architecture for UAM and two opposing architecture options were compared: a centralized collision avoidance architecture and a decentralized collision avoidance architecture. The benefits and tradeoffs that were identified demonstrated that this architecture development framework enables a comparison of architecture options that is more focused on control-related benefits and tradeoffs.

Based on these benefits and tradeoffs, a preferred collision avoidance architecture was chosen for UAM. Because UAM flights are expected to be on demand, traffic circumstances are expected to be more unpredictable. Thus, a shared collision avoidance architecture was proposed for UAM because it provides the necessary flexibility to enable the ATM system to adapt its behavior to the prevailing air traffic circumstances as they change.

The second design iteration then focused on refining this high-level shared collision avoidance architecture developed in iteration 1 to obtain a more detailed definition of that architecture. Thus, the architecture development framework was applied again to refine both the conceptual architecture and the system architecture for implementing shared collision avoidance. The results from both iterations demonstrated that this architecture development framework can be applied iteratively to incrementally refine an ATM architecture for UAM. This is achieved through the iterative use of STPA analyses and the structured processes provided by the framework that enables systems engineers to make more informed early architectural design decisions driven by safety considerations.

Finally, this research also developed a supporting framework for identifying the assumptions underlying design decisions made during architecture development and ensuring that they remain valid over time. This supporting framework included guiding prompts to help systems engineers consider any assumptions they might be making at each step of architecture development. These assumptions can then be used to generate either system requirements or assumptions-based leading indicators to account for these assumptions in downstream design decisions and monitor their validity over time. This supporting framework was demonstrated for the UAM case study to show how different types of assumptions about the ATM system or the airspace environment can be identified at each step of architecture development

The remainder of this chapter summarizes each of the three research contributions, discusses their limitations, and describes potential avenues for future work.

7.1 Contribution 1: Safety-Relevant Criteria for Comparing Architecture Options

The first contribution of this research is that this architecture development framework supports the identification of safety-relevant evaluation criteria for comparing architecture options and this was demonstrated in Chapter 4. Two main parts of the framework enable this. First, the framework provides a process for performing an STPA scenario-based comparison of the architecture options. By analyzing each architecture option using STPA and then comparing the identified scenarios across the different options, analysts or systems engineers can more easily determine the control-related behavioral differences between the architecture options and the architectural elements (e.g., control actions, feedback) that give rise to those differences.

Second, once these behavioral differences have been identified, this architecture development framework then provides a structure for generating evaluation criterion – a short phrase that describes a control-related difference in behavior between the architecture options. Thus, the evaluation criteria are qualitative and highlight the aspects of an architecture option's behavior that contribute to better or worse control behavior.

The structure of an evaluation criterion consists of four main parts, each of which provides control-relevant information to support the comparison of architecture options:

1. **Characteristic**: An attribute (e.g., responsiveness, timeliness) of the control behavior being described
2. **Control Aspect**: The aspect of control being described – Decision making, Process models, Feedback and control inputs, or Control path
3. **Hazard:** The hazard that the control behavior being described is intended to control
4. **Scenario Context**: The context under which the control behavior being described occurs

This structure not only helps guide an analyst or systems engineer in generating an evaluation criterion, but it also ensures better consistency and uniformity across the various evaluation criteria, especially when different criteria are generated by different people. Based on these evaluation criteria, benefits and tradeoffs can be more easily identified for each architecture option. These benefits and tradeoffs can then be used to inform decisions about what architecture would best achieve the desired emergent properties.

To evaluate if the evaluation criteria generated by this framework are relevant for comparing architecture options in terms of emergent properties like safety, two opposing ATM architecture options were compared in design iteration 1 of the case study: (1) a centralized collision avoidance architecture and (2) a decentralized collision avoidance architecture. The benefits and tradeoffs identified using this framework were then compared to those found by similar studies conducted in the existing literature. This comparison found that the evaluation criteria identified by this framework can identify more control-related benefits and tradeoffs that cover more areas of control. This gives an analyst or systems engineer a broader understanding of how the various control-related aspects of an architecture option contribute to better or worse control behavior. The benefits and tradeoffs are also more focused on the control-related differences between architecture options, and it is easier to identify the architectural elements that give rise to those differences. In addition, the benefits and tradeoffs are derived from a broader consideration of different air traffic contexts. These findings therefore provide support for *Hypothesis 1: A systems-theoretic approach can identify relevant criteria for comparing architecture options and evaluating their ability to achieve emergent properties.*

There are several limitations that must be acknowledged for this part of the work. First, the comparison of benefits and tradeoffs against existing literature is subject to some author bias. As discussed at the beginning of Section 4.4, most of the benefits and tradeoffs of centralized and decentralized architectures that were identified by the existing literature were quantitative whereas those identified by this framework are qualitative. Thus, the author had to apply engineering judgement to interpret the quantitative results in the existing literature and determine the implied qualitative benefit or tradeoff. However, every effort was made to ensure that any qualitative benefit or tradeoff that could reasonably have been identified based on the quantitative results were included in the comparison.

Another limitation of this part of the work is that comparing STPA scenarios across architecture options to generate the evaluation criteria depends heavily on the analyst being familiar with the behavior of each architecture option. This is because the analyst needs to be able to determine how each architecture might behave under a given scenario. Although the control structure model for each architecture option is available, the comparison process does not describe how to use it to more carefully consider the behavioral differences between architecture options. Future work could therefore consider developing a more systematic process for deriving or generating the evaluation criterion that makes better use of the control structures for each architecture option to better support identifying the control-related behavioral differences between architecture options.

The third limitation of this part of the work is that the formulation of evaluation criteria can still vary significantly depending on the analyst who is generating it. While the structure for formulating evaluation criteria helps to reduce some of that variability, it does not eliminate it.

In addition, the example characteristics provided in this dissertation were not intended to be exhaustive. Future work could therefore consider performing a more thorough characterization of the different types of control attributes that are commonly seen in systems. Such a characterization would provide a more comprehensive set of attributes to help systems engineers generate a more consistent set of evaluation criterion.

Finally, the work in this dissertation focused on generating the evaluation criteria but did not consider prioritization of those criteria. Especially when numerous evaluation criteria are identified, being able to prioritize the evaluation criteria would help analysts and systems engineers to focus on the most important benefits and tradeoffs when comparing architecture options and deciding what the preferred system architecture should be. Future work could therefore consider how to prioritize the evaluation criteria. For example, this prioritization could be informed by the priority order of the associated hazards (or losses).

One other possible direction of future work is to consider different ways in which the evaluation criteria can be grouped and analyzed to identify architectural patterns – assignments of responsibilities that consistently give rise to favorable system behavior. In this dissertation, evaluation criteria were grouped and analyzed by control aspect, but other groupings are possible that could generate other types of insights. For example, the evaluation criteria could be grouped by hazard or by context to identify differences in behavior associated with a particular hazard or differences in behavior in a certain context.

7.2 Contribution 2: Structured Processes for Developing the System Architecture

The second contribution of this research is that this architecture development framework provides appropriate types of support for developing and refining both the behavior of a system and its system architecture. This was demonstrated in Chapter 5 and two main aspects of the framework enable this. First, the framework makes use of iterative STPA analyses to provide safety-relevant information about the behavior of the conceptual architecture (in the behavioral design process) or the architecture options (in the structural design process) as they are created. This iterative analysis of the system using STPA allows analysts or systems engineers to incrementally learn about how the behavior of the system evolves as they make design decisions.

Second, the framework provides structured processes for using the STPA analysis results to support making informed design decisions. In this architecture development framework, there are three main types of design decisions that a systems engineer makes:

1. Decisions to **identify the control elements** needed to create the conceptual architecture
2. Decisions to **create architecture options** based on the responsibility assignments that can mitigate or prevent unsafe behavior
3. Decisions to **develop the preferred system architecture** based on the control-related benefits and tradeoffs identified for the various architecture options

Thus, using the structured processes provided by this framework, each of these design decisions are informed by what is needed to prevent unsafe or undesirable behavior that are identified using STPA.

To evaluate if these processes provide the necessary support to systems engineers to enable them to iteratively refine a system architecture, the results from the two design iterations performed for the UAM case study were reviewed. This review demonstrated that these structured processes provided by this framework do enable incremental refinement of the ATM architecture for UAM. The first design iteration started with an abstract control structure of the ATM system and developed a shared collision avoidance architecture as the preferred architecture for UAM. Then, the second design iteration refined this shared collision avoidance architecture to define more specific control responsibilities for safely managing shared collision avoidance. Architecture options were then considered for how to implement one of these more specific control responsibilities in the ATM system architecture. Thus, these results show that the structured processes discussed earlier in this section enable systems engineers to make more informed early design decisions driven by safety considerations.

In addition, traceability is maintained between the various design artifacts created using this architecture development framework. This traceability not only enables forward refinement of the system architecture to incrementally add detail to it, but it also allows systems engineers to reconsider and revise past design decisions when needed. These findings therefore provide support for *Hypothesis 2: A systems-theoretic approach can support making informed design decisions to iteratively develop and refine the architecture for a system.*

There are several limitations that must be acknowledged for this part of the work. First, although the goal of the structural design process is to incrementally improve the system architecture by exploring different architecture options, incremental improvements do not necessarily always lead to identifying the best system architecture. This is because the responsibilities defined in the conceptual architecture do not behave independently of one another. As a result, the relationship between responsibility assignments and the resulting system behavior is not monotonic. Even if earlier design iterations identify incrementally better architecture options, it is possible that later iterations identify and evaluate architecture options that were thought to be better but ultimately show significantly worse behavior than those evaluated in prior iterations.

Another related limitation is that the comparison of architecture options performed using this framework does not allow an analyst or systems engineer to create an absolute rank ordering of the architecture options (e.g., from best to worst). This is because the comparisons performed using this framework only generate relative differences between architecture options, not absolute differences. Although these comparisons might allow an analyst or systems engineer to rank architecture options compared in the same iteration, architecture options from different iterations cannot be ordered the same way without performing additional analyses and comparing those additional results.

The third limitation of this part of the work is that although this dissertation has demonstrated that the framework enables architecture refinement, there is no guarantee that this refinement can continue all the way to detailed system design and allow systems engineers to generate a physical architecture. This is because this work was primarily focused on early-stage architecture development where the system architecture is primarily functional. Physical components are essentially only represented in terms of groups of assigned functions or responsibilities. Thus, one avenue for future work is to perform further design refinement using this framework to determine the extent to which the architecture of a system can be incrementally refined.

Finally, one other limitation of this part of the work is that it can be challenging to keep track of the STPA results between iterations and maintain traceability between abstract UCAs and scenarios and the more detailed UCAs and scenarios. This can be especially difficult when large numbers of UCAs and scenarios are generated for multiple losses and hazards. In addition, when design decisions are revised, it can be difficult to determine which STPA results are affected by the revision. Thus, another avenue for future work could investigate how to organize and keep track of the results obtained from different iterations of STPA analyses and make it easier to determine how revising a design decision impacts the STPA analyses that have already been conducted.

The refinement that is enabled by this architecture development framework provides one other interesting direction for future work. By refining a system architecture using this framework, a systems engineer can design the refined version of an architecture to meet the goals of the more abstract version. For example, in the case study described in this dissertation, the refined ATM architecture developed in design iteration 2 was designed to meet the goal of implementing the shared collision avoidance architecture developed in design iteration 1.

This suggests that this type of refinement could be applied to create an architecture-based certification process. For example, a regulator could be responsible for developing the system architecture up to a certain level of abstraction to establish the requirements and goals for that system. Then, individual companies (e.g., vehicle manufacturers) could continue refining the architecture to create their own implementation of the higher-level architecture created by the regulator. Certification of those individual implementations would then involve regulators verifying that each lower-level refined architecture is consistent with the abstract architecture and meets all its requirements.

7.3 Contribution 3: Identifying and Accounting for Underlying Assumptions

The last contribution of this research is the development of a supporting framework for identifying and accounting for underlying assumptions during architecture development. This supporting framework was developed because it is important to ensure that any assumptions underlying design decisions made during architecture development remain valid both during system development and once the system is placed into operation. This is because assumptions that are invalidated could lead to flaws in the system design. Thus, the goal of this supporting framework is to help systems engineers identify any assumptions they are making as they develop a system architecture and account for them as architecture development proceeds.

This supporting framework was developed in Chapter 6 and consisted of two main parts. First, guiding prompts that are tailored to each step of the architecture development framework help systems engineers consider what must be true about the system or the environment for the system to behave as intended. Second, once the underlying assumptions have been identified, they can be monitored and accounted for in the system design. For both assumptions about the environment or the system, assumptions-based leading indicators can be generated to monitor their validity over time, especially during operations. In addition, assumptions about the system can be used to derive additional system requirements that the system will have to meet. This ensures that downstream design decisions do not violate assumptions made earlier in the design process. Furthermore, the validity of these system assumptions can be verified at the end of development using the verification and validation process for a system because a verified derived requirement implies a validated assumption.

Although no evaluation was performed for this part of the work, this supporting framework was demonstrated for the UAM case study. The case study results showed how underlying assumptions about both the environment and system could be identified throughout the development process including during each of the STPA analyses, the generation of requirements, the development of the conceptual architecture and the comparison of architecture options.

There are several limitations that should be acknowledged for this part of the work. First, only a demonstration of this supporting framework was performed, and no evaluation of this supporting framework was done. However, such an evaluation could be done in the future using data about assumptions identified using an alternate process to compare to the assumptions identified by this supporting framework.

Another limitation is that the identification of underlying assumptions is done only using guiding questions as prompts instead of being supported by an analysis of the system or the design decision being made. Thus, future work could investigate a more rigorous method of analysis to more systematically capture underlying assumptions instead of only relying on guiding prompts. Especially for assumptions about the system, this analysis method could help analysts or systems engineers identify assumptions that are more closely linked to specific elements or behavioral aspects of an architecture or control structure.

Finally, a third limitation of this part of the work is that the guiding prompts were not intended to be a complete or exhaustive list of things to consider when identifying underlying assumptions. In addition, these guiding prompts were developed based only on the case study in this dissertation. As a result, the broader applicability of the guiding prompts to other types of systems has not been evaluated and is likely limited. Thus, future work could perform a more rigorous characterization of the types of assumptions made when developing different types of systems. This characterization could then be used to develop more general guidance on the types of assumptions to look for at each step of architecture development.

Abbreviations and Acronyms

ABP	Assumptions-Based Planning
ATM	Air Traffic Management
ATC	Air Traffic Control
ConOps	Concept of Operations
DSM	Design Structure Matrix
FAA	Federal Aviation Administration
IFR	Instrument Flight Rules
IMLEO	Initial Mass to Low Earth Orbit
INCOSE	International Council on Systems Engineering
MBSE	Model-Based Systems Engineering
MCAS	Maneuvering Characteristics Augmentation System
MVC	Model/View/Controller Framework
NAS	National Airspace System
NASA	National Aeronautics and Space Administration
OOSEM	Object-Oriented Systems Engineering Methodology
PBSE	Pattern-Based Systems Engineering
RA	Resolution Advisory
SDADF	Safety-Driven Architecture Development Framework
STECA	Systems-Theoretic Early Concept Analysis
STAMP	Systems Theoretic Accident Model and Processes
STPA	Systems Theoretic Process Analysis
SysML	Systems Modeling Language
TA	Traffic Advisory
TCAS	Traffic Collision Avoidance System
TFR	Temporary Flight Restriction
UAM	Urban Air Mobility
UCA	Unsafe Control Action
UCCA	Unsafe Collaborative Control Action

Glossary

Term	Definition
Architecture Option	One possible way to assign the responsibilities (and their associated control actions and feedback) to either existing or new controllers in the system.
Architecture Tradespace	The set of all possible assignments of responsibilities defined in the conceptual architecture to controllers in the system.
Assignment Constraints	A preferred responsibility assignment that could mitigate or eliminate a causal scenario that was identified using STPA.
Causal/Loss Scenario	A description of the causal factors that can lead to the unsafe control actions and to the hazards [89, p. 42].
Conceptual Architecture	A functional control structure that models the desired control behavior of a system in terms of the required responsibilities, control actions, and feedback.

Term	**Definition**
Conflict *(e.g., between aircraft)*	A situation where there is a risk for collision between aircraft and/or vehicles [115].
Constraint Requirement	A system requirement that describes restrictions on acceptable ways that a control decision should be made or the expected response of the controlled process (used to derive responsibility constraints).
Control Requirement	A system requirement that describes a control decision or control function that needs to be performed (used to derive responsibilities)
Evaluation Criteria	A short phrase describing a control-related difference in behavior between the architecture options being compared.
Hazard	A system state or set of conditions that, together with a particular set of worst-case environmental conditions, will lead to a loss [89, p. 17].
Loss	Something of value to stakeholders [89, p. 16].
Process Model Parts	Information needed by a controller to make appropriate decisions when carrying out a responsibility.
Responsibility	A control function to be performed or a control decision that needs to be made to enforce a safety constraint.
Responsibility Constraint	A restriction on how the associated responsibility should be performed.
Safety Constraint *(also referred to as a system-level constraint in [89])*	A specification of system conditions or behaviors that need to be satisfied to prevent hazards (and ultimately prevent losses) [89, p. 20].
System Architecture	An abstract description of the entities of a system and the relationships between them [9, p. 2].
Unsafe Control Action (UCA)	A control action that, in a particular context and worst-case environment, will lead to a hazard [89, p. 35].

Bibliography

[1] S. Friedenthal *et al.*, "Systems Engineering Vision 2035: Engineering Solutions for a Better World," International Council on Systems Engineering (INCOSE), 2021.

[2] N. Leveson, *Engineering a safer world: systems thinking applied to safety*. in Engineering systems. Cambridge, Mass: MIT Press, 2011.

[3] Aptiv *et al.*, "Safety First for Automated Driving," 2019. [Online]. Available: https://group.mercedes-benz.com/documents/innovation/other/safety-first-for-automated-driving.pdf

[4] Garmin, "Garmin Autonomí® | Autonomous Flight Solutions." Accessed: Sep. 21, 2024. [Online]. Available: https://discover.garmin.com/en-US/autonomi/

[5] Wisk Aero, "Concept of Operations: Autonomous UAM Aircraft Operations and Vertiport Integration," Apr. 2022.

[6] NASA, "Terrain Relative Navigation: Landing Between the Hazards." Accessed: Sep. 21, 2024. [Online]. Available: https://science.nasa.gov/science-research/science-enabling-technology/technology-highlights/terrain-relative-navigation-landing-between-the-hazards/

[7] United Nations Committee on the Peaceful Uses of Outer Space, "Guidelines for the Long-Term Sustainability of Outer Space Activities," AC.105/2018/CRP.20, Jun. 2018. Accessed: Jan. 10, 2023. [Online]. Available: https://www.unoosa.org/res/oosadoc/data/documents/2018/aac_1052018crp/aac_1052018crp_20_0_html/AC105_2018_CRP20E.pdf

[8] P. Checkland, *Systems thinking, systems practice*. Chichester [Sussex] ; New York: J. Wiley, 1999.

[9] E. Crawley *et al.*, "The influence of architecture in engineering systems," *MIT Engineering Systems Monograph*, 2004.

[10] E. F. Crawley, B. Cameron, and D. Selva, *System architecture: strategy and product development for complex systems*. Boston: Pearson, 2016.

[11] S. Friedenthal, H. Lykins, and A. Meilich, "Adapting UML for an Object Oriented Systems Engineering Method (OOSEM)," in *Proceedings of the INCOSE 2000 International Symposium*, Minneapolis, MN, Jul. 2000.

[12] H.-P. Hoffmann, "SysML-based systems engineering using a model-driven development approach," *White Paper, Telelogic*, 2008.

[13] P. Y. Papalambros and D. J. Wilde, *Principles of optimal design modeling and computation*, 3rd editon. Cambridge, United Kingdom ; New York, NY 10006, USA: Cambridge University Press, 2017.

[14] N. Leveson, "An Improved Design Process for Complex Control-Based Systems Using STPA and a Conceptual Architecture," MIT, White Paper, 2019.

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

[16] N. Leveson, “A systems approach to risk management through leading safety indicators,” *Reliability Engineering & System Safety*, vol. 136, pp. 17–34, Apr. 2015, doi: 10.1016/j.ress.2014.10.008.

[17] B. Hill *et al.*, “UAM Vision Concept of Operations (ConOps) UAM Maturity Level (UML) 4,” NASA, Dec. 2020. [Online]. Available: https://ntrs.nasa.gov/citations/20205011091

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

[19] 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.

[20] P. D. Vascik, R. J. Hansman, and N. S. Dunn, “Analysis of Urban Air Mobility Operational Constraints,” *Journal of Air Transportation*, vol. 26, no. 4, pp. 133–146, Oct. 2018, doi: 10.2514/1.D0120.

[21] “Fast Forwarding to a Future of On-Demand Urban Air Transportation,” Uber Elevate, Oct. 2016. Accessed: Jul. 02, 2022. [Online]. Available: https://evtol.news/__media/PDFs/UberElevateWhitePaperOct2016.pdf

[22] B. Lascara, A. Lacher, M. DeGarmo, D. Maroney, R. Niles, and L. Vempati, “Urban Air Mobility Airspace Integration Concepts,” The MITRE Corporation, Jun. 2019.

[23] P. D. Vascik, H. Balakrishnan, and R. J. Hansman, “Assessment of air traffic control for urban air mobility and unmanned systems,” presented at the 8th International Conference for Research in Air Transportation, 2018.

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

[25] “Systems Engineering for Intelligent Transportation Systems: An Introduction for Transportation Professionals.” Department of Transportation, Office of Operations, Jan. 2007.

[26] D. D. Walden, G. J. Roedler, K. Forsberg, R. D. Hamelin, T. M. Shortell, and International Council on Systems Engineering, Eds., *Systems engineering handbook: a guide for system life cycle processes and activities*, 4th edition. Hoboken, New Jersey: Wiley, 2015.

[27] U.S. Department of Defense, “The DOD Architecture Framework Version 2.02.” Accessed: Jan. 06, 2023. [Online]. Available: https://dodcio.defense.gov/library/dod-architecture-framework/

[28] D. Hornford, N. Hornford, M. Lambert, and K. Street, “An Introduction to the TOGAF Standard, 10th Edition,” Apr. 2022. Accessed: Jan. 06, 2023. [Online]. Available: https://pubs.opengroup.org/architecture/w212/?_ga=2.243256962.1008969126.1673023210-829819966.1673023210

[29] L. Delligatti, *SysML distilled: a brief guide to the systems modeling language*. Upper Saddle River, NJ: Addison-Wesley, 2014.

[30] D. Garlan and M. Shaw, “An Introduction to Software Architecture.” Jan. 1994.

[31] G. D. Bergland, “A guided tour of program design methodologies,” *Computer*, vol. 14, no. 10, pp. 13–37, 1981.

[32] K. T. Ulrich, S. D. Eppinger, and M. C. Yang, *Product design and development*, Seventh edition. New York, NY: McGraw-Hill Education, 2020.

[33] “ISO 26262:2018 Road Vehicles - Functional Safety.” International Standards Organization (ISO), Dec. 2018.

[34] “ISO 21448:2022 Road Vehicles - Safety of the Intended Functionality.” International Standards Organization (ISO), Jun. 2022.

[35] D. L. Parnas, “On the criteria to be used in decomposing systems into modules,” in *Pioneers and their contributions to software engineering*, Springer, 1972, pp. 479–498.

[36] D. L. Parnas, “Designing software for ease of extension and contraction,” *IEEE transactions on software engineering*, no. 2, pp. 128–138, 1979.

[37] N. Wirth, *Systematic programming: an introduction*. in Prentice-Hall series in automatic computation. Englewood Cliffs, N.J: Prentice-Hall, 1973.

[38] “ISO/IEC/IEEE 15288 Systems and software engineering - System Lifecycle Processes,” IEEE.

[39] NASA, *NASA Systems Engineering Handbook*, rev2 ed. Place of publication not identified: 12TH MEDIA SERVICES, 2017.

[40] J. A. Estefan, “Survey of model-based systems engineering (MBSE) methodologies,” *Incose MBSE Focus Group*, vol. 25, no. 8, pp. 1–12, 2007.

[41] D. Dori, *Object-process methodology: a holistic systems paradigm*, Softcover repr. of the hardcover 1. ed. 2002. Berlin: Springer, 2013.

[42] “The Rational Unified Process for Systems Engineering: A Rational Software White Paper.” Rational Software Corporation, 2001.

[43] L. Baker Jr and J. E. Long, “Role of System Engineering Across The System Life Cycle,” *Vitech white paper, Vitech Corporation, Vienna, VA*, 2000.

[44] M. D. Ingham, R. D. Rasmussen, M. B. Bennett, and A. C. Moncada, “Generating requirements for complex embedded systems using State Analysis,” *Acta Astronautica*, vol. 58, no. 12, pp. 648–661, Jun. 2006, doi: 10.1016/j.actaastro.2006.01.005.

[45] C. H. Fleming and N. G. Leveson, “Improving hazard analysis and certification of integrated modular avionics,” *Journal of Aerospace Information Systems*, vol. 11, no. 6, pp. 397–411, 2014.

[46] S. D. Eppinger and T. R. Browning, *Design structure matrix methods and applications*. in Engineering systems. Cambridge, Mass. London: MIT Press, 2012.

[47] H.-P. Hoffmann, “Systems Engineering Best Practices with the Rational Solution for Systems and Software Engineering: Deskbook Release 4.1,” IBM Corporation, 2011. Accessed: Jul. 07, 2022. [Online]. Available: https://jazz.net/library-content/wp-content/uploads/2020/11/ibm_rational_harmony_deskbook_rel_4.1.pdf

[48] E. Rechtin, *Systems architecting: creating and building complex systems*. Englewood Cliffs, N.J: Prentice Hall, 1991.

[49] E. Gamma, R. Helm, R. Johnson, and J. Vlissides, *Design patterns: elements of reusable object-oriented software*. in Addison-Wesley professional computing series. Reading, Mass: Addison-Wesley, 1995.

[50] B. Schindel, "INCOSE/OMG MBSE Patterns Working Group," OMG Standards Development Organization MBSE Wiki. Accessed: Jan. 19, 2023. [Online]. Available: https://www.omgwiki.org/MBSE/doku.php?id=mbse:patterns:patterns

[51] R. E. Johnson, "Components, frameworks, patterns," *SIGSOFT Softw. Eng. Notes*, vol. 22, no. 3, pp. 10–17, May 1997, doi: 10.1145/258368.258378.

[52] R. E. Johnson, "Frameworks = (components + patterns)," *Commun. ACM*, vol. 40, no. 10, pp. 39–42, Oct. 1997, doi: 10.1145/262793.262799.

[53] W. D. Schindel and T. Peterson, "An Overview of Pattern-Based Systems Engineering (PBSE): Leveraging MBSE Techniques," presented at the INCOSE Enchantment Chapter Webinar, May 14, 2014.

[54] B. G. Cameron and E. F. Crawley, "Crafting platform strategy based on anticipated benefits and costs," in *Advances in product family and product platform design*, Springer, 2014, pp. 49–70.

[55] C. Zhang and D. Budgen, "What do we know about the effectiveness of software design patterns?," *IEEE Transactions on Software Engineering*, vol. 38, no. 5, pp. 1213–1231, 2011.

[56] N. G. Leveson and K. A. Weiss, "Making embedded software reuse practical and safe," in *Proceedings of the 12th acm sigsoft twelfth international symposium on foundations of software engineering*, 2004, pp. 171–178.

[57] D. Selva and E. F. Crawley, "VASSAR: Value assessment of system architectures using rules," in *2013 IEEE Aerospace Conference*, IEEE, 2013, pp. 1–21.

[58] B. H. Y. Koo, "A Meta-language for Systems Architecting," Massachusetts Institute of Technology, Cambridge, MA, 2005.

[59] W. L. Simmons, "A framework for decision support in systems architecting," PhD Thesis, Massachusetts Institute of Technology, 2008.

[60] N. P. Suh, *Axiomatic design: advances and applications*. in The MIT-Pappalardo series in mechanical engineering. New York: Oxford University Press, 2001.

[61] J. Agte, O. de Weck, J. Sobieszczanski-Sobieski, P. Arendsen, A. Morris, and M. Spieck, "MDO: assessment and direction for advancement—an opinion of one international group," *Struct Multidisc Optim*, vol. 40, no. 1–6, pp. 17–33, Jan. 2010, doi: 10.1007/s00158-009-0381-5.

[62] J. R. R. A. Martins and S. A. Ning, *Engineering design optimization*. Cambridge ; New York, NY: Cambridge University Press, 2021.

[63] R. Dechter, *Constraint Processing*. San Francisco: Morgan Kaufmann Publishers, 2003.

[64] B. Williams, "Constraint Programming: Problem and Propagation," presented at the 16.410 Lecture, Fall 2022, Oct. 31, 2022.

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

[66] 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.

[67] M. A. Kammoun, N. Rezg, and Z. Achour, “New approach for air traffic management based on control theory,” *International Journal of Production Research*, 2014.

[68] P. K. Menon, G. D. Sweriduk, and K. D. Bilimoria, “New approach for modeling, analysis, and control of air traffic flow,” *Journal of guidance, control, and dynamics*, 2004.

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

[70] 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.

[71] B. F. Gore, B. L. Hooey, E. Mahlstedt, and D. C. Foyle, “Evaluating NextGen Closely Spaced Parallel Operations concepts with validated human performance models: Scenario development and results,” NASA Technical Report NASA/TM-2013-216503, 2013.

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

[73] 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.

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

[75] 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.

[76] 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. doi: 10.1109/DASC.2002.1052917.

[77] S. Lee, A. Pritchett, and D. Goldsman, “Hybrid agent-based simulation for analyzing the national airspace system,” in *Proceeding of the 2001 Winter Simulation Conference*, IEEE, 2001.

[78] M. W. Maier and E. Rechtin, *The art of systems architecting*, 3rd ed. Boca Raton: CRC Press, 2009.

[79] F. DeRemer and H. H. Kron, “Programming-in-the-Large Versus Programming-in-the-Small,” *IIEEE Trans. Software Eng.*, vol. SE-2, no. 2, pp. 80–86, Jun. 1976, doi: 10.1109/TSE.1976.233534.

[80] T. Ishimatsu, O. De Weck, J. Hoffman, Y. Ohkami, and R. Shishko, “A generalized multi-commodity network flow model for space exploration logistics,” in *AIAA SPACE 2013 Conference and Exposition*, 2013, p. 5414.

[81] O. De Weck, “Our Future on Earth and in Space: Engineering Systems as dynamic Generalized Multi-Commodity Network Flow Systems,” presented at the Course 16.89 Guest Lecture, Apr. 06, 2022.

[82] E. Yourdon and L. L. Constantine, *Structured design: fundamentals of a discipline of computer program and systems design*. Englewood Cliffs, N.J: Prentice Hall, 1979.

[83] L. von Bertalanffy, *General System Theory: Foundations, Development, Applications*. New York: George Braziller Inc, 1968.

[84] G. M. Weinberg, *An introduction to general systems thinking: Gerald M. Weinberg*, Silver anniversary ed. New York: Dorset House, 2001.

[85] K. Whitney, J. M. Bradley, D. E. Baugh, and C. W. C. Jr, “Systems theory as a foundation for governance of complex systems,” *International Journal of System of Systems Engineering*, vol. 6, no. 1–2, pp. 15–32, 2015.

[86] P. M. Senge, *The fifth discipline: the art and practice of the learning organization*, Rev. and Updated. New York: Doubleday/Currency, 2006.

[87] K. M. Adams, P. T. Hester, J. M. Bradley, T. J. Meyers, and C. B. Keating, “Systems theory as the foundation for understanding systems,” *Systems Engineering*, vol. 17, no. 1, pp. 112–123, 2014.

[88] J. Sterman, *Business dynamics: systems thinking and modeling for a complex world*. Boston: Irwin/McGraw-Hill, 2000.

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

[90] H. M. Slominski, “Using STPA and CAST to Design for Serviceability and Diagnostics,” Masters, Massachusetts Institute of Technology, Cambridge, MA, 2020.

[91] A. Kharsansky, “A systemic approach toward scalable, reliable and safe satellite constellations,” MIT System Design and Management Program, Masters Thesis, Sep. 2020.

[92] M. E. France, “Engineering for Humans: A New Extension to STPA,” MIT Department of Aeronautics and Astronautics, Masters Thesis, Jun. 2017.

[93] D. Horney, “Systems-Theoretic Process Analysis and Safety-Guided Design of Military Systems,” Masters, MIT, 2017.

[94] C. H. Fleming, “Safety-Driven Early Concept Analysis and Development,” MIT Department of Aeronautics and Astronautics, PhD Dissertation, Feb. 2015.

[95] A. N. Kopeikin, N. G. Leveson, and N. A. Neogi, “Defining Collaborative Control Interactions Using Systems Theory,” *INCOSE International Symp*, vol. 33, no. 1, pp. 895–909, Jul. 2023.

[96] J. Krozel, M. Peters, K. D. Bilimoria, C. Lee, and J. S. B. Mitchell, “System Performance Characteristics of Centralized and Decentralized Air Traffic Separation Strategies,” *Air Traffic Control Quarterly*, vol. 9, no. 4, pp. 311–332, Oct. 2001, doi: 10.2514/atcq.9.4.311.

[97] J. Hoekstra, R. Van Gent, and R. Ruigrok, “Conceptual design of free flight with airborne separation assurance,” in *Guidance, Navigation, and Control Conference and Exhibit*, Boston,MA,U.S.A.: American Institute of Aeronautics and Astronautics, Aug. 1998. doi: 10.2514/6.1998-4239.

[98] M. Ballin, J. Hoekstra, D. Wing, and G. Lohr, "NASA Langley and NLR research of distributed air/ground traffic management," in *AIAA's Aircraft Technology, Integration, and Operations (ATIO) 2002 Technical Forum*, 2002, p. 5826.

[99] C. Tomlin, G. J. Pappas, and S. Sastry, "Conflict resolution for air traffic management: A study in multiagent hybrid systems," *IEEE Transactions on automatic control*, vol. 43, no. 4, pp. 509–521, 1998.

[100] Federal Aviation Administration, "Urban Air Mobility Concept of Operations v2.0," Apr. 2023.

[101] J. M. Hoekstra, R. N. H. W. van Gent, and R. C. J. Ruigrok, "Designing for safety: the 'free flight' air traffic management concept," *Reliability Engineering & System Safety*, vol. 75, no. 2, pp. 215–232, Feb. 2002, doi: 10.1016/S0951-8320(01)00096-5.

[102] P. H. Abreu, E. Oliveira, A. Camara, and D. Silva, "Comparing a centralized and decentralized multi-agent approaches to air traffic control," in *Proceedings of the 28th European Simulation and Modelling Conference, Porto, Portugal*, 2014, pp. 189–193.

[103] M. Xue, "Urban Air Mobility Conflict Resolution: Centralized or Decentralized?," presented at the AIAA Aviation 2020, 2020.

[104] H. A. Blom, G. J. Bakker, B. Klein Obbink, and M. B. Klompstra, "Free flight safety risk modelling and simulation," 2006, Accessed: Jun. 28, 2024. [Online]. Available: https://reports.nlr.nl/bitstream/10921/340/1/TP-2006-290.pdf

[105] D. Wing *et al.*, "Comparison of ground-based and airborne function allocation concepts for NextGen using human-in-the-loop simulations," in *10th AIAA Aviation Technology, Integration, and Operations (ATIO) Conference*, 2010, p. 9293.

[106] H. A. P. Blom and G. J. Bakker, "Safety Evaluation of Advanced Self-Separation Under Very High En Route Traffic Demand," *Journal of Aerospace Information Systems*, vol. 12, no. 6, pp. 413–427, Jun. 2015, doi: 10.2514/1.I010243.

[107] Federal Aviation Administration, "Introduction to TCAS II, Version 7.1." Federal Aviation Administration, Feb. 28, 2011.

[108] German Federal Bureau of Aircraft Accidents Investigation, "Investigation Report," German Federal Bureau of Aircraft Accidents Investigation, May 2004.

[109] B. Wong, "A STAMP model of the Überlingen aircraft collision accident," Thesis, Massachusetts Institute of Technology, 2004. Accessed: Sep. 05, 2024. [Online]. Available: https://dspace.mit.edu/handle/1721.1/28861

[110] N. G. Leveson, "Intent specifications: an approach to building human-centered specifications," *IEEE Transactions on Software Engineering*, vol. 26, no. 1, pp. 15–35, Jan. 2000.

[111] J. A. Dewar, *Assumption-based planning: a tool for reducing avoidable surprises*. in RAND studies in policy analysis. Cambridge ; New York: Cambridge University Press, 2002.

[112] H. W. Rittel and M. M. Webber, "Dilemmas in a general theory of planning," *Policy sciences*, vol. 4, no. 2, pp. 155–169, 1973.

[113] R. O. Mason and I. I. Mitroff, *Challenging strategic planning assumptions: theory, cases, and techniques*. New York: Wiley, 1981.

[114] National Transportation Safety Board, “Assumptions Used in the Safety Assessment Process and the Effects of Multiple Alerts and Indications on Pilot Performance,” Safety Recommendation Report ASR-19-01, Sep. 2019.

[115] “Conflict Solving,” *Skybrary*. [Online]. Available: https://skybrary.aero/articles/conflict-solving#:~:text=A%20situation%20where%20there%20is,Source%3A%20ICAO

Appendix A Design Iteration 1 – Initial STPA Analysis of NAS

This appendix contains the STPA results for the initial analysis of the NAS that was performed at the beginning of design iteration 1 (shown in Section 4.1). For this STPA analysis, the losses, hazards, and control structure were already presented in Table 7, Table 8, and Figure 24 respectively. This appendix presents the full set of UCAs and scenarios that were generated for the abstract *Coordination* control action in that control structure. In Table A-1 and Table A-2, UCAs for which causal scenarios were generated are highlighted in blue.

A.1 Unsafe Control Actions (UCAs) for *Coordination* Control Action

Table A-1: Providing and not providing UCAs for Coordination Control Action

Not Providing	Providing
UCA-1.1: Air Traffic Management does not coordinate the interaction between two UAM aircraft or a UAM aircraft and another airspace user when a collision between them is imminent *[H-1, H-3]*	**UCA-1.15:** Air Traffic Management coordinates air traffic to allow UAM aircraft to access the airspace when the NAS does not have sufficient capacity *[H-1, H-3, H-4]*
UCA-1.2: Air Traffic Management does not coordinate air traffic in the airspace to assist UAM aircraft in an emergency *[H-1, H-2, H-3]*	**UCA-1.16:** Air Traffic Management coordinates the movement of UAM aircraft when UAM aircraft need to access an airport and that coordination interferes with approach/arrival traffic for a nearby airport *[H-1, H-3, H-6]*
UCA-1.3: Air Traffic Management does not coordinate the movement of UAM aircraft when UAM aircraft are interfering with approach/arrival traffic for a nearby airport *[H-1, H-3]*	**UCA-1.17:** Air Traffic Management coordinates air traffic to give UAM aircraft access to the airspace when UAM aircraft do not meet the necessary criteria for access to an airspace *[H-1, H-2, H-3, H-6]*
UCA-1.4: Air Traffic Management does not coordinate air traffic to allow UAM aircraft to access the airspace when UAM aircraft need to execute a mission and meet the criteria for access to that airspace *[H-3]*	**UCA-1.18:** Air Traffic Management coordinates UAM aircraft when their flight paths will result in excessive environmental effect *[H-4]*
UCA-1.5: Air Traffic Management does not coordinate the movements of UAM aircraft when they are about to fly into a section of airspace where air traffic must be excluded (e.g., for safety or security reasons) *[H-5]*	**UCA-1.19:** Air Traffic Management coordinates UAM aircraft such that they interfere with the operations of other NAS users *[H-3, H-6]*
UCA-1.6: Air Traffic Management does not coordinate UAM air traffic to reduce the number of aircraft using the airspace when it exceeds the capacity of Air Traffic Management to coordinate them *[H-1, H-3, H-4, H-5]*	**UCA-1.20:** Air Traffic Management provides coordination that UAM aircraft are not fully capable of executing *[H-1, H-2, H-3, H-6]*

Not Providing	Providing
UCA-1.7: Air Traffic Management does not coordinate UAM air traffic when demand for a particular part of the airspace exceeds acceptable levels of use *[H-4]*	**UCA-1.21:** Air Traffic Management provides coordination that causes a collision with an obstacle or terrain *[H-1, H-2, H-3, H-5]*
UCA-1.8: Air Traffic Management does not coordinate the movements of UAM aircraft when they interfere with the operations of other NAS users *[H-1, H-3]*	**UCA-1.22:** Air Traffic Management provides coordination that causes a collision with another aircraft *[H-1, H-2, H-3]*
UCA-1.9: Air Traffic Management does not coordinate UAM aircraft when their operation has an excessive environmental effect *[H-4]*	**UCA-1.23:** Air Traffic provides coordination when two UAM aircraft are departing or arriving at the same location at the same time *[H-1, H-3, H-5, H-6]*
UCA-1.10: Air Traffic Management does not coordinate UAM aircraft when they need to access a conventional airport *[H-3]*	**UCA-1.24:** Air Traffic Management provides coordination that causes unnecessary or unacceptable operational impacts (e.g., delays) to the flight *[H-3, H-6]*
UCA-1.11: Air Traffic Management does not coordinate UAM aircraft when two aircraft want access to the same location at the same time *[H-3]*	**UCA-1.25:** Air Traffic Management provides coordination when that coordination leads the aircraft toward inclement weather that could interfere with flight operations *[H-1, H-2, H-3, H-5]*
UCA-1.12: Air Traffic Management does not coordinate UAM aircraft when they are close to an obstacle or terrain *[H-1]*	**UCA-1.26:** Air Traffic Management provides coordination to restrict UAM flights when the restrictions cause travel time using UAM to increase beyond acceptable levels *[H-3]*
UCA-1.13: Air Traffic Management does not coordinate UAM aircraft when inclement weather is approaching that could interfere with flight operations *[H-1, H-2, H-3]*	**UCA-1.27:** Air Traffic Management provides coordination to UAM traffic that forces them to use airspace where the ride is unpleasant or unsafe for passengers *[H-2, H-6]*
UCA-1.14: Air Traffic Management does not coordinate UAM aircraft when congestion has increased beyond acceptable levels *[H-1, H-3, H-4]*	**UCA-1.28:** Air Traffic Management provides coordination to UAM aircraft that does not satisfy priority needs (e.g., an aircraft running out of fuel needs access to an airport sooner than one that has plenty of fuel) *[H-1, H-2, H-3]*

Table A-2: Too early/late and applied too long/stopped too soon UCAs for Coordination Control Action

Too Early / Too Late	Applied Too Long / Stopped Too Soon
UCA-1.29: Air Traffic Management coordinates the interaction between two UAM aircraft or a UAM aircraft and another airspace user too late to prevent violation of minimum separation between them *[H-1, H-2, H-3]*	**UCA-1.35:** Air Traffic Management provides coordination to UAM aircraft in the airspace too long when conditions have changed such that the coordination provided is no longer valid *[H-1, H-2, H-3, H-4, H-6]*
UCA-1.30: Air Traffic Management coordinates UAM aircraft too late to assist them in an emergency *[H-1, H-2, H-3]*	**UCA-1.36:** Air Traffic Management stops coordinating air traffic in the airspace too soon before the emergency experienced by UAM aircraft is resolved *[H-1, H-2, H-3]*
UCA-1.31: Air Traffic Management coordinates air traffic to allow UAM aircraft access to the airspace too late after the time window in which UAM aircraft need that access *[H-3]*	**UCA-1.37:** Air Traffic Management stops coordinating UAM aircraft too soon to prevent UAM aircraft from entering a restricted section of airspace when air traffic still needs to be excluded from that section of airspace *[H-2, H-4, H-6]*
UCA-1.32: Air Traffic Management coordinates UAM aircraft too late after environmental effects of UAM operations have exceeded acceptable levels *[H-4]*	**UCA-1.38:** Air Traffic Management stops coordinating UAM aircraft too soon before environmental effects of system operation have returned to acceptable levels *[H-4]*
UCA-1.33: Air Traffic Management provides coordination too late after congestion has exceeded acceptable levels *[H-1, H-3, H-4, H-6]*	**UCA-1.39:** Air Traffic Management restricts air traffic for too long after environmental effects of system operation have returned to acceptable levels *[H-3]*
UCA-1.34: Air Traffic Management provides coordination too late after UAM aircraft interfere with the operations of another airspace user *[H-3, H-6]*	**UCA-1.40:** Air Traffic Management provides coordination to restrict UAM flights for too long after congestion is within acceptable levels but travel time remains unacceptable or service consistency remains unacceptable *[H-3]*
	UCA-1.41: Air Traffic Management stops providing coordination too soon when there is pressure to allow more flights to take place but UAM cannot be safely operated with a higher traffic density *[H-1, H-3, H-4, H-6]*

A.2 Causal Scenarios for Selected UCAs

This section shows the causal scenarios that were identified for select UCAs highlighted in blue in Table A-1 and Table A-2. Note that each scenario is also traced to the system requirement generated to mitigate or prevent it using the requirement links in square braces included at the end of each requirement.

<u>**Scenarios for UCA-1.1:**</u> Air Traffic Management does not coordinate the interaction between two UAM aircraft or a UAM aircraft and another airspace user when a collision between them is imminent *[H-1, H-3]*

CS-1.1.1. Air Traffic Management does not provide coordination when a collision is imminent. Air Traffic Management has received feedback of the potential conflict, but does not issue coordination because:

CS-1.1.1-1. The Air Traffic Management believes at least one of the aircraft is a false positive and therefore ignores the feedback and wrongly believes that a collision is not actually imminent [↓ Req-1, Req-2]

CS-1.1.1-2. Alternatively, Air Traffic Management is pre-occupied (either in the human or automated sense) with other tasks and does not have the capacity to recognize or handle the potential collision and provide coordination to prevent it [↓ Req-3, Req-4]

CS-1.1.1-3. The Air Traffic Management wrongly believes that the aircraft involved have already been provided coordination and therefore wrongly believes that it does not need to provide further coordination to prevent the collision [↓ Req-5]

CS-1.1.1-4. The Air Traffic Management believes at least one of the aircraft will recognize the potential collision and change its path to avoid the collision and therefore the Air Traffic Management wrongly believes that it does not need to provide coordination to prevent the collision [↓ Req-5]

CS-1.1.1-5. The Air Traffic Management is unable to select an acceptable coordination solution because the environment is constrained and there are no options available to the Air Traffic Management to coordinate the aircraft that does not cause another violation of minimum separation [↓ Req-6]

CS-1.1.2. Air Traffic Management does not receive feedback of the potential conflict because:

CS-1.1.2-1. Equipment used to detect and identify aircraft in the airspace has failed or is delayed and either only partial information about an aircraft is received by Air Traffic Management or no information at all is received by Air Traffic Management [↓ Req-4, Req-7]

CS-1.1.2-2. There are more aircraft in the airspace than Air Traffic Management is capable of detecting and tracking simultaneously. As a result, it receives incomplete feedback about the aircraft present in the airspace. [↓ Req-8]

CS-1.1.2-3. Equipment used by the Air Traffic Management to detect and track aircraft is insufficiently performant (e.g., insufficient resolution or update rate). As a result, information regarding the position, speed or intent of the aircraft are inaccurate or incomplete. Alternatively, certain parts of that information might be missing. In either case, the Air Traffic Management therefore does not receive all the information needed to recognize potential conflict. [↓ Req-4]

CS-1.1.2-4. Traffic data is manipulated such that at least one of the aircraft is removed and therefore the Air Traffic Management is unaware of the presence of that aircraft and therefore does not recognize the potential conflict [↓ Req-9]

CS-1.1.2-5. The Air Traffic Management is not aware of future intended movements of the aircraft (e.g., about to turn left into the path of another aircraft) and wrongly assumes that the aircraft will continue on their current trajectories. Air Traffic Management therefore wrongly believes that a collision is not imminent. [↓ Req-10, Req-11]

CS-1.1.2-6. The Air Traffic Management has wrong or out-of-date information about the future intended movements from either or both aircraft and wrongly believes based on that intent information that a collision is not imminent [↓ Req-11]

CS-1.1.3. Air Traffic Management provides coordination when a collision is imminent. However, that control is not received by the aircraft because:

CS-1.1.3-1. The method for communicating that coordination to the aircraft has failed, is unavailable or is degraded by environmental conditions [↓ Req-12, Req-13]

CS-1.1.3-2. The coordination channel is over capacity and Air Traffic Management is unable to transmit its coordination to the aircraft [↓ Req-3, Req-5, Req-12, Req-13]

CS-1.1.3-3. The communication channel used by Air Traffic Management to transmit its coordination does not match the channel that the aircraft is listening on to receive that coordination [↓ Req-13]

CS-1.1.3-4. Air Traffic Management transmits coordination to the wrong aircraft and therefore the intended aircraft does not receive that coordination [↓ Req-14]

CS-1.1.3-5. Air Traffic Management transmits coordination to the correct aircraft but a different aircraft wrongly believes the coordination is for them and executes the coordination [↓ Req-13]

CS-1.1.4. Air Traffic Management provides coordination when a collision is imminent. The coordination is received by the aircraft but it is not effective in preventing violation of minimum separation because:

CS-1.1.4-1. The aircraft is unable to execute the coordination provided by Air Traffic Management to avoid violation of minimum separation. This might occur if:

CS-1.1.4-1.1. The coordination provided by Air Traffic Management exceeds the capabilities of the aircraft [↓ Req-15]

CS-1.1.4-1.2. The aircraft is preoccupied with another task and does not attempt to execute the coordination before the aircraft violates minimum separation. It may also occur if the provided coordination is incorrect or insufficient for resolving the conflict. [↓ Req-5, Req-12]

CS-1.1.4-1.3. The aircraft might believe that coordination provided by the Air Traffic Management would result in another violation of minimum separation and therefore choose to ignore the Air Traffic Management's coordination and make an independent decision which results in a violation of minimum separation anyway [↓ Req-6, Req-17]

CS-1.1.4-1.4. The coordination provided by Air Traffic Management resolves the original imminent collision but causes another violation of minimum separation [↓ Req-6, Req-17]

CS-1.1.4-2. There is insufficient time after the Air Traffic Management provides coordination for the aircraft to execute the coordination to avoid violation of minimum separation [↓ Req-18]

CS-1.1.4-3. The aircraft believes that it has executed the coordination even though it has not actually done so. [↓ Req-5]

CS-1.1.4-4. The aircraft receives the coordination but deliberately chooses to ignore it (e.g., hijacking or other malicious activity) [↓ Req-80]

<u>**Scenarios for UCA-1.8:**</u> Air Traffic Management does not coordinate the movements of UAM aircraft when they interfere with the ability of other NAS users to achieve their missions [H-4]

CS-1.8.1. Air Traffic Management has received feedback that UAM aircraft are interfering with the operations of other NAS users but does not issue coordination because:

CS-1.8.1-1. Air Traffic Management is preoccupied addressing higher priority tasks (e.g., assisting an aircraft experiencing an emergency) and does not have the capacity to provide coordination to reduce the impact of UAM aircraft on other NAS users [↓ Req-3, Req-4]

CS-1.8.1-2. Air Traffic Management wrongly believes that UAM aircraft's impact on other NAS users is negligible or tolerable by the other NAS users and therefore there is no need to issue coordination to reduce the impact [↓ Req-20, Req-21]

CS-1.8.1-3. Air Traffic Management does not recognize that the operation of UAM aircraft is negatively affecting a mission to fulfill a public benefit (e.g., search & rescue, medevac, public safety operations) and instead believes that the mission only fulfills commercial/private interests and therefore wrongly decides to allow the UAM aircraft to interfere with the operation of the other NAS users without providing coordination [↓ Req-20, Req-21]

CS-1.8.1-4. Air Traffic Management does not realize that another NAS user has a time-critical mission to execute (e.g., cargo that needs to be delivered to the destination by a certain time) or a time-critical need (e.g., running out of fuel and cannot maintain a hold) and believes that the other NAS user's mission can be delayed for the UAM aircraft and therefore wrongly decides not to provide coordination to avoid the delay for the other NAS user [↓ Req-20, Req-21]

CS-1.8.1-5. Air Traffic Management is unable to find a coordination solution that avoids interference with the operations of other NAS users because there is insufficient capacity in the system to prevent NAS users affecting each other's operations [↓ Req-3, Req-66]

CS-1.8.1-6. Air Traffic Management is told by federal regulators to prioritize the UAM aircraft over other NAS users and the Air Traffic Management therefore chooses to ignore the feedback and does not issue coordination to avoid interference with the other NAS users [↓ Req-20, Req-21, Req-22, Req-39]

CS-1.8.1-7. Air Traffic Management has incorrect information about the flight plans and acceptable interference limits of other airspace users and therefore do not realize that a UAM flight will interfere with it [↓ Req-11, Req-21]

CS-1.8.2. Air Traffic Management has not received feedback that UAM aircraft are interfering with the operations of other NAS users because:

CS-1.8.2-1. Air Traffic Management does not have sufficient information about the mission or intentions of other NAS users or the UAM aircraft to know that UAM aircraft are interfering with their operations. [↓ Req-11]

CS-1.8.2-2. Air Traffic Management does not have sufficiently performant detection or reporting mechanisms to identify instances when UAM aircraft are interfering with the operations of other NAS users [↓ Req-22]

CS-1.8.2-3. Detection or reporting mechanisms available to Air Traffic Management only report interferences with a delay large enough that by the time Air Traffic Management receives feedback that UAM aircraft are interfering with the operations of other NAS users, the interference is no longer occurring [↓ Req-4, Req-22]

CS-1.8.2-4. The impact to operations of other NAS users occurs slowly/gradually or there is a small impact to a large number of NAS users and Air Traffic Management does not receive feedback about the overall extent of the impact to the operations of other NAS users [↓ Req-22]

CS-1.8.2-5. Air Traffic Management does not receive direct feedback about interference and only uses UAM congestion level as their measure of whether the flight operations of other airspace users might be impacted by UAM flights. Thus, when other airspace users are impacted while UAM congestion is below threshold, Air Traffic Management wrongly believes there is no need to issue coordination [↓ Req-22]

CS-1.8.3. Air Traffic Management provides coordination when UAM aircraft interfere with the operations of other NAS users. *Scenarios are similar to those for CS-1.1.3.*

CS-1.8.4. Air Traffic Management provides coordination to prevent a UAM aircraft from interfering with the operations of other NAS users. The coordination is received by UAM aircraft but is not effective in preventing interference because:

CS-1.8.4-1. The Air Traffic Management identifies or is alerted to the impending interference at the last minute and does not provide the coordination with enough time for UAM aircraft to respond before they interfere with the operation of other NAS users [↓ Req-18, Req-22, Req-23]

CS-1.8.4-2. Air Traffic Management changes its coordination to avoid an interference and provides that coordination at the last minute. [↓ Req-18, Req-23]

<u>**Scenarios for UCA-1.14:**</u> Air Traffic Management does not coordinate UAM aircraft when congestion has increased beyond acceptable levels *[H-1, H-3, H-4]*

CS-1.14.1. Air Traffic Management receives feedback that congestion has increased beyond acceptable levels but does not coordinate UAM aircraft because:

CS-1.14.1-1. Air traffic management believes that although congestion has increased beyond acceptable levels, the accident risk has not increased because UAM aircraft are also improving their collision avoidance capabilities. As a result, they believe that they do not need to coordinate UAM aircraft to prevent a collision. [↓ Req-36, Req-37]

CS-1.14.1-2. Although congestion has exceeded threshold levels along certain routes, Air Traffic Management wrongly believe and assume that UAM operators will gradually reroute aircraft along other routes to reduce congestion. As a result, they do not issue coordination themselves [↓ Req-30, Req-39]

<u>**Scenarios for UCA-1.32:**</u> Air Traffic Management coordinates UAM aircraft too late after environmental effects of UAM operations have exceeded acceptable levels *[H-4]*

CS-1.32.1. Air Traffic Management received feedback on time indicating that environmental effects have exceeded acceptable levels but provide coordination too late because:

CS-1.32.1-1. They have the wrong mental model of the acceptable level of environmental effect and wrongly believe that the environmental effect is still acceptable. As a result, they do not issue coordination to limit further increase in environmental effects [↓ Req-25]

CS-1.32.1-2. Although they recognize that the environmental effect has been exceeded, they wrongly believe that UAM operators will curb further increase in environmental effects themselves and therefore do not take any action themselves [↓ Req-30, Req-31]

CS-1.32.1-3. Air Traffic Management wrongly believes that UAM traffic will reduce soon (e.g., peak period will end, surge will subside) and therefore believes that a momentary exceedance of acceptable environmental effects can be tolerated [↓ Req-27]

CS-1.32.1-4. Congestion is not at unsafe limits and ridership is high. As such, Air Traffic Management attempts to continue to allow flights to depart, hoping to meet as much of the demand for flights as possible. As a result, they do not start to restrict flights to curb environmental effects until they have exceeded acceptable levels [↓ Req-27]

CS-1.32.2. Air Traffic Management does not receive feedback that the environmental effect of UAM operations have exceeded acceptable levels because

CS-1.32.2-1. There is a delay reporting these environmental effects to Air Traffic Management (e.g., reports must be manually made by community members). As a result, by the time Air Traffic Management is aware of these reports, acceptable levels have already been exceeded [↓ Req-25]

CS-1.32.2-2. Air Traffic Management only receives feedback when levels have been exceeded and therefore is unable to take action before levels have been exceeded [↓ Req-26]

CS-1.32.2-3. Air Traffic Management has erroneous data about the current state of the airspace and the future intent of aircraft and therefore have the wrong mental model of what the anticipated environmental effect of UAM will be in the future. As a result, they do not issue coordination to limit the environmental impact of UAM operations until the tolerable threshold has already been exceeded [↓ Req-7]

Scenarios for UCA-1.33: Air Traffic Management provides coordination too late after congestion has exceeded acceptable levels *[H-1, H-3, H-4, H-6]*

CS-1.33.1. Air Traffic Management has received feedback that congestion has exceeded acceptable levels but provides coordination too late after congestion has exceeded acceptable levels because:

CS-1.33.1-1. There is pressure from UAM operators not to restrict flights to avoid increasing the level of ride sharing amongst passengers. As a result, Air Traffic Management chooses to wait to impose restrictions on flights [↓ Req-28, Req-29]

CS-1.33.1-2. Air Traffic Management believes that the demand surge (e.g., caused by rush hour, a major sporting event) will shortly subside and therefore believes traffic volume will reduce without requiring additional intervention [↓ Req-30, Req-31]

CS-1.33.1-3. Air Traffic Management has the wrong mental model of the threshold congestion at which coordination is needed and therefore wrongly believes that congestion needs to worsen further before they need to provide coordination [↓ Req-26, Req-31, Req-32]

CS-1.33.1-4. Air Traffic Management is busy coordinating non-UAM aircraft and does not process the feedback showing UAM congestion has increased beyond acceptable levels until the congestion level has exceeded acceptable levels [↓ Req-4]

CS-1.33.2. Air Traffic Management does not receive feedback indicating that congestion has exceeded acceptable levels on time because:

CS-1.33.2-1. Data indicating the level of UAM congestion is reported to Air Traffic Management with a delay. As a result, Air Traffic Management does not recognize that coordination is needed to reduce the level of congestion until congestion has exceeded acceptable levels [↓ Req-1]

Scenarios for UCA-1.34: Air Traffic Management provides coordination too late after UAM aircraft interfere with the operations of another airspace user [H-3, H-6]

CS-1.34.1. Air Traffic Management receives feedback that indicated that UAM aircraft are/will interfere with the operations of other airspace users on time. However, they provide coordination too late because:

CS-1.34.1-1. The other airspace user (e.g., a public safety flight, private jet flight) is also conducting a short-notice/on-demand operation and Air Traffic Management does not have sufficient time to provide adequate coordination to prevent the impact of UAM aircraft on their operation before it occurs [↓ Req-41, Req-42]

CS-1.34.1-2. The other airspace user changes their flight plan at the last minute such that Air Traffic Management does not have sufficient time to provide adequate coordination to prevent the impact of UAM aircraft on their operation before it occurs [↓ Req-41, Req-42]

CS-1.34.1-3. Air Traffic Management receives feedback of the interference with very short notice to when the interference will occur (e.g., because the UAM flight was being performed on-demand). As a result, there is insufficient time to make a coordination decision before the flight is performed and the interference occurs [↓ Req-23]

<u>Scenarios for UCA-1.41:</u> Air Traffic Management stops providing coordination too soon when there is pressure to allow more flights to take place but UAM cannot be safely operated with a higher traffic density *[H-1, H-3, H-4, H-6]*

CS-1.41.1. Air Traffic Management received feedback on time that UAM cannot be safely operated with a higher traffic density but still decide to stop coordinating aircraft because:

CS-1.41.1-1. Air Traffic Management believes that UAM traffic density will soon decrease significantly (e.g., as rush hour ends) and therefore believes that allowing a temporary rise in UAM flights to reduce delays is permissible [↓ Req-30]

CS-1.41.1-2. Air Traffic Management assumes that unexpected incidents or emergencies will not arise and believes they can handle a higher density of UAM aircraft. However, when an emergency arises, they are unable to safely coordinate all of the aircraft while addressing the emergency [↓ Req-34]

CS-1.41.2. Air Traffic Management does not receive feedback that indicates that UAM cannot be safely operated with a higher traffic density because:

CS-1.41.2-1. Under pressure to improve profitability and serve more passengers, UAM operators provide feedback to Air Traffic Management that indicates that the capabilities of their aircraft are sufficient for operating at a higher traffic density even though they are not. Without verifying this feedback, Air Traffic Management uses it to and update its process model of the level of UAM traffic that UAM operators can handle safely make its decisions [↓ Req-35]

Appendix B Design Iteration 1 – Requirements and Control Elements

In this appendix, the system requirements and control elements that were identified to create the iteration 1 conceptual architecture (shown in Figure 28) are presented. In addition, any underlying assumptions associated with the requirements or control elements are included. Note that this appendix shows just the final set of requirements and control elements that were identified after several iterations and does not show how the requirements and control elements evolved between iterations.

B.1 NAS System-Level Collision Avoidance Requirements

Req-1. ATM system shall be able to track all aircraft in the airspace to ensure sufficient separation [↓ RC-1]

System Assumption: Assumes it is possible to achieve tracking performance of <relevant minimum tracking performance specifications>

Req-2. ATM system shall verify erroneous detections within <TBD time> before choosing to ignore them [↓ RC-74]

Environment Assumption: Assumes that flights are known within <TBD> time of desired departure

System Assumption: There will always be at least 1 alternative option for tracking aircraft in the airspace that can be used to verify a suspected erroneous detection [Req-46]

Req-3. ATM system shall ensure that sufficient capacity is available to detect and coordinate all aircraft that have or will need access to the airspace [↓ Resp-2]

Environment Assumption: Assumes that surges in demand for flights will occur with at least <TBD mins> of advance notice for the NAS to implement plans to mitigate system impacts

System Assumption: Assumes that Req-8 is also carried out at the same time whenever demand nears capacity limits [Req-61]

Req-4. ATM system shall coordinate the movement of aircraft to resolve any potential conflicts either between two aircraft or a conflict of an aircraft trajectory with terrain [↓ Resp-1 (iteration 1), Resp-1.1 (iteration 2)]

Environment Assumption: Assumes that flights are known within <TBD> time of desired departure

System Assumption: Assumes coordination decisions can be made within <TBD time> [Req-49]

System Assumption: Assumes that there is coordination with Req-3 to ensure sufficient capacity is available to manage the current or anticipated future level of traffic [Req-48]

Req-5. ATM system shall ensure that aircraft that need coordination have received coordination, are executing it correctly and that the risk of collision or interference is no longer present [↓ RC-6 (iteration 1), Resp-1.4 (iteration 2)]

System Assumption: Assumes that there is coordination between this requirement and Req-4, and Req-27 to ensure that coordination is effective [Req-31, Req-50]

Req-6. ATM system shall ensure that acceptable coordination options are always available for aircraft to avoid violation of minimum separation [↓ Resp-3]

<u>System Assumption</u>: Assumes that there is sufficient airspace available (low enough density) to allow alternative movement options to be established [Req-51]

<u>System Assumption</u>: Assumes that there is coordination with Req-8 to manage airspace usage to ensure sufficient airspace is available for this [Req-52]

Req-7. ATM system shall detect when information needed to identify and track aircraft is missing, delayed, erroneous or not available and take action to restore that information [↓ RC-75]

<u>System Assumption</u>: Alternative options are available for obtaining tracking information or checking the status of tracking equipment [Req-46]

<u>System Assumption</u>: Assumes there is coordination with Req-4 to account for information being out-of-date when making coordination decisions [Req-53]

Req-8. ATM system shall only allow as many users to access the airspace as it is capable of detecting, tracking and coordinating [↓ Resp-4]

<u>System Assumption</u>: Assumes Req-3 is also performed to manage capacity (e.g., during surge times) [Req-47]

Req-9. ATM system shall prevent the manipulation or tampering of data used for detecting and tracking aircraft [↓ RC-19]

<u>System Assumption</u>: Assumes Req-3 is also performed to manage capacity (e.g., during surge times) [Req-47]

Req-10. ATM system shall account for intended movements of aircraft in addition to current trajectories to detect potential collisions [↓ RC-2]

<u>Environment Assumption</u>: Assumes that aircraft are willing to share their intended trajectories for at least <TBD time> into the future (e.g., no privacy concerns)

Req-11. ATM system shall ensure that information about the intent, mission, acceptable operational impacts and future intended movements of aircraft is available, does not contain errors and is kept updated [↓ RC-32]

<u>Environment Assumption</u>: Assumes users are willing to share mission and intent information

<u>System Assumption</u>: Assumes there is coordination with Req-4 to account for intent information being out-of-date when making coordination decisions [Req-56]

Req-12. ATM system shall coordinate the movements of other aircraft to prevent violation of minimum separation with an aircraft that is unable to communicate or not responding [↓ RC-15]

<u>System Assumption</u>: Assumes coordination with Req-6 where the availability of alternative movement options is already being assured [Req-57]
<u>System Assumption</u>: Assumes that when it is discovered that an aircraft is unable to communicate, new coordination decisions can be made within <TBD> time to resolve any imminent collisions [Req-58]

Req-13. ATM system shall ensure that aircraft have received the coordination being communicated [↓ RC-26]

Req-14. ATM system shall ensure that coordination is communicated to the correct aircraft [↓ RC-117]

Req-15. ATM system shall ensure that coordination provided to the aircraft is within the capabilities of the aircraft [↓ RC-3]

Req-17. ATM system shall ensure that coordination provided to the aircraft does not cause another violation of minimum separation [↓ RC-4]

Req-18. ATM system shall provide coordination that allows for and accounts for delays due to response time of <TBD> to enact the coordination [↓ RC-5]

Req-20. ATM system shall consider access priorities when issuing coordination or managing access to the airspace [↓ RC-118]

Req-21. ATM system shall account for any users' constraints on mission execution in addition to access priorities to determine which impacts to operations are acceptable when coordinating aircraft [↓ RC-7]

Req-22. ATM system shall notify users if their operations will be impacted beyond acceptable [↓ RC-43]

Req-23. ATM system shall be able to detect any unexpected operational impacts experienced by an airspace user within <TBD> time of the interference occurring [↓ RC-24]

Req-24. ATM system shall respond to impending interference and issue coordination instructions within TBD period of time [↓ RC-8]

<u>Environment Assumption</u>: Assumes that flights are known within <TBD> time of desired departure

<u>System Assumption</u>: Assumes that flight operations can respond to last-minute coordination within <TBD> time [Req-18]

Req-25. ATM system shall establish and maintain acceptable levels of noise and visual pollution levels as well as emissions levels

Req-26. ATM system shall be able to monitor environmental effects with acceptable levels of performance to enable high levels of environmental effects to be detected before the exceedance occurs [↓ RC-17]

Req-27. ATM system shall be able to make preemptive coordination decisions based on trends in environmental effects and congestion to help prevent acceptable thresholds from being exceeded

<u>System Assumption</u>: It is assumed that this requirement is performed in coordination with Req-4 (coordination for collision avoidance) [Req-55]

<u>System Assumption</u>: It is assumed that thresholds defined in Req-25 and Req-32 are used in this requirement [Req-33]

Req-28. ATM system shall prioritize flight safety over UAM passenger safety if both cannot be assured [↓ RC-9]

<u>Environment Assumption</u>: This assumes that UAM operators will be able to work with other NAS stakeholders to manage passenger safety even if flights need to be restricted to maintain airspace safety

Req-29. ATM system shall ensure that ride sharing does not exceed levels necessary to ensure safety of UAM riders

Req-30. If anticipated conditions are used to make coordination decisions, ATM system shall confirm that the anticipated conditions do occur

<u>System Assumption</u>: Assumes that if anticipated conditions do not occur within <TBD time>, coordination decisions are re-evaluated via Req-4 and Req-27 [Req-31]

Req-31. If anticipated conditions do not occur within <TBD> time, ATM system shall re-evaluate its coordination decisions [↓ RC-37]

Req-32. ATM system shall establish and maintain clear and measurable specifications for the threshold congestion that is acceptable

Req-33. ATM system shall make use of threshold congestion and environmental effects when making preemptive coordination decisions [↓ RC-16]

Req-34. Air Traffic Management must always have some amount of reserve capacity set aside to provide additional coordination during unexpected emergencies or incidents [↓ RC-21]

Req-35. ATM system shall consider the capabilities of the UAM aircraft and operators (e.g., pilot training, aircraft equipage etc) when establishing or re-evaluating threshold congestion levels [↓ RC-39]

Req-36. ATM system shall ensure that all aircraft have the capabilities required for new threshold congestion levels when new congestion thresholds are introduced [↓ SR-21]

System Assumption: Assumes there is coordination with changes to congestion level thresholds to ensure they are rolled out synchronously [Req-62]

Req-37. ATM system shall ensure that current congestion threshold levels are enforced even if some aircraft are capable of operations at higher congestion levels [↓ RC-38]

Req-38. ATM system shall ensure that any event that will result in significant operational impacts to airspace users is communicated to users with <TBD> advanced notice so that airspace users can adjust their plans [↓ RC-35]

Req-39. If assuming that another aircraft or controller will take an action when deciding on coordination, ATM system shall confirm with the aircraft or controller that the action will be taken prior to implementing coordination [↓ RC-36]

Req-40. ATM system shall manage both demand for flights in addition to the ability of flights to access the airspace when mitigating congestion and vehicle sharing

System Assumption: Assumes that this requirement is coordinated with Req-8 so that demand or airspace access is managed both by managing how many flights are needed and how many flights can be accepted [Req-63]

Req-41. ATM system shall ensure that UAM aircraft abide by the TFR associated with the public safety events to ensure that UAM aircraft do not interfere with public safety flights

Environment Assumption: Assumes that public safety events will continue to be accompanied by a TFR that will ensure that aircraft stay away and therefore avoid interference
System Assumption: This is coordinated with Req-4 (for collisions) and Req-27 (for avoiding negative environmental or congestion effects) [Req-64]

Req-42. ATM system shall establish a minimum notification window within which avoidance of operational impact cannot be guaranteed [↓ RC-42]

Req-43. ATM system shall ensure that routes of flight used by UAM aircraft minimize time spent over residential neighborhoods and other community spaces, especially during periods when occupancy is high [↓ RC-18]

Environment Assumption: Assumes that the concerns that the public would have against UAM (e.g., noise, visual pollution, public safety) would stem from UAM operating around neighborhoods when occupancy is high

Req-44. At key choke/convergence points (e.g., airports, vertiports), ATM system shall ensure that the UAM aircraft do not interfere with conventional air traffic flights

Environment Assumption: Assumes that regular UAM passenger flights will be required to work around scheduled commercial aircraft since they have fixed schedules known well in advance and serve larger quantities of passengers with each flight
Environment Assumption: Assumes that the main areas in which interference between UAM and conventional air traffic might occur is at/around airports and vertiports

Req-45. ATM system shall ensure that capacity determinations account for both traffic density and coverage area [↓ RC-22]

Req-46. ATM system shall have at least 2 options for tracking aircraft (current position, speed, heading, ID) in the airspace to verify erroneous detections [↓ RC-20]

Req-47. ATM system shall ensure that aircraft trajectories do not consume more airspace than is reasonable to allocate for that aircraft

Req-48. ATM system shall coordinate between capacity management and traffic management to ensure that sufficient capacity exists to manage the current or anticipated future level of traffic [↓ RC-12]

Req-49. ATM system shall be able to make coordination decisions within <TBD> time [↓ RC-10]

Req-50. ATM system shall ensure that if coordination was not effective, coordination is evaluated again to ensure that collision risks are adequately mitigated [↓ RC-27]

Req-51. ATM system shall ensure that there is sufficient airspace available (or low enough density) available to allow alternative movement options to be established [↓ RC-25]

Req-52. ATM system shall ensure that access to the airspace is managed in accordance with what is needed to ensure that acceptable coordination options are always available [↓ RC-28]

Req-53. ATM system shall ensure that coordination decisions account for whether information is missing/out-of-date [↓ RC-29]

Req-55. ATM system shall ensure that coordination issued to aircraft consider both potential future environmental impact as well as more immediate conflict avoidance [↓ RC-11]

Req-56. ATM system shall ensure that coordination decisions account for whether intent information is missing/out-of-date [↓ RC-30]

Req-57. ATM system shall ensure that alternative movement options can be used to coordinate aircraft [↓ RC-31]

Req-58. ATM system shall ensure that coordination decisions can be made within <TBD> time after an aircraft is discovered to be unable to communicate to resolve any imminent collisions [↓ RC-13]

Req-59. ATM system shall ensure that there are at least two methods for communicating coordination with aircraft [↓ RC-14]

Req-61. ATM system shall modify how aircraft trajectories are modified, alternative trajectories are selected and airspace access is managed based on an initiated traffic management program [↓ RC-54]

Req-62. ATM system shall ensure that new congestion thresholds are coordinated with ensuring that aircraft have the capabilities needed for those new congestion thresholds to ensure that they are rolled out synchronously [↓ RC-40]

Req-63. ATM system shall coordinate between management of flight demand and management or airspace access [↓ RC-41]

Req-64. ATM system shall ensure that UAM aircraft abide by public safety event TFRs while also avoiding collisions and negative environmental effects or congestion [↓ RC-42]

Req-65. ATM system shall require regular UAM passenger flights to work around scheduled commercial aircraft [↓ RC-34]

Req-66. ATM system shall ensure that unexpected operational impacts are detected and mitigated where necessary to prevent them from occurring again

Req-68. ATM system shall grant an aircraft experiencing an emergency the highest priority access to the airspace they need to address the emergency [↓ RC-44]

Req-69. ATM system shall ensure that there is enough spare airspace available to keep other aircraft away from a non-communicative aircraft [↓ RC-45]

Req-70. When making preemptive coordination decisions to prevent environmental or congestion exceedances, ATM system shall manage capacity and airspace access in addition to issuing modified trajectories [↓ RC-46]

Req-71. ATM system shall ensure that erroneous detections are made known to Resp-1 so that conflict resolution accounts for errors in detections [↓ RC-47]

Req-72. ATM system shall use an alternative location source for tracking aircraft that does not suffer from the same inaccuracy limitations as the primary source [↓ RC-48]

Req-73. ATM system shall manage generate alternate trajectories and manage air traffic in accordance with ATM capacity using a consolidated view of the airspace [↓ RC-23]

Req-74. ATM system shall recompute alternative movement options within <TBD> time so that re-evaluations can happen continuously [↓ RC-50]

Req-75. ATM system shall monitor the movements of aircraft that are unable to communicate to ensure they are behaving as expected [↓ RC-51]

Req-76. ATM system shall ensure that any new capacity expansion or airspace access management plans are implemented [↓ RC-52]

Req-77. ATM system shall coordinate ride demand management with congestion and environmental effects management [↓ RC-53]

Req-78. ATM system shall ensure that the overall operational impact incurred by an aircraft is considered and minimized when making coordination decisions [↓ RC-59]

Req-79. ATM system shall inform airspace users if they will be significantly affected by a TFR [↓ RC-55]

Req-80. ATM system shall be able to prevent an aircraft that is not communicating and/or disobeying coordination instructions from causing damage or harm to people or property on the ground

Req-81. ATM system shall coordinate between ensuring behavior of aircraft matches issued coordination and addressing aircraft behaving erratically [↓ RC-56]

Req-82. ATM system shall coordinate between ride demand and flight dispatch to ensure ride demand is coordinated with flight dispatch [↓ RC-57]

Req-83. ATM system shall ensure that any proposed coordination has new alternative trajectories available before issuing the proposed coordination [↓ RC-58]

Req-84. If a trajectory modification is not effective at resolving the collision, the reason for the modification not being effective must be determined so that an updated trajectory modification can account for it [↓ RC-60]

Req-85. ATM system shall account for reasons that a trajectory modification was ineffective when selecting new trajectory modifications [↓ RC-61]

Req-86. ATM system shall check in with affected aircraft on preferred trajectory modification if unable to meet all operational constraints

Req-87. ATM system shall ensure that tracking and trajectory information is available for all aircraft within <TBD distance> of the UAM operating environment [↓ RC-62]

Req-88. ATM system shall ensure that tracking and trajectory information is available for all aircraft before the aircraft has entered the UAM operating environment [↓ RC-63]

Req-89. ATM system shall ensure that a conflict-free trajectory is available for an aircraft (either from the ground or in the air) prior to allowing it to enter UAM airspace [↓ RC-64]

Req-90. ATM system shall monitor and confirm that an aircraft is following its planned trajectory to the accuracy specified with that trajectory [↓ RC-65]

Req-91. ATM system shall re-evaluate an aircraft's trajectory and issue trajectory modifications if needed if an aircraft deviates from its planned trajectory by more than <TBD> [↓ RC-66]

Req-92. ATM system shall ensure that all trajectory modifications are transmitted and acknowledged within <TBD> time [↓ RC-67]

<u>System Assumption</u>: Assumes that if trajectory modifications are not acknowledged within <TBD> time, the conflict associated with that modification will be flagged for re-evaluation [Req-93]

Req-93. ATM system shall re-evaluate trajectory modification(s) associated with a conflict if the trajectory modification(s) are not acknowledged within <TBD> time [↓ RC-68]

Req-94. ATM system shall provide accompanying navigation accuracy and expected response time parameters when deciding trajectory modifications to ensure navigation accuracy and response time expectations are made explicit [↓ RC-69]

Req-95. ATM system shall ensure that any changes to relevant operational constraints are accounted for when issuing trajectory modifications [↓ RC-70]

Req-96. ATM system shall ensure that any tall obstacles of at least <TBD> in height that could interfere with flight operations have their presence and duration (if temporary) reported and disseminated to aircraft and operators

Req-97. ATM system shall discuss other flight plan options (e.g., earlier departure, different arrival/departure aerodrome) with aircraft if an excessive operational constraint will be incurred [↓ RC-72]

Req-98. ATM system shall account for anticipated weather and potential future air traffic needs in addition to already filed flights when making trajectory modifications [↓ RC-49]

Req-99. ATM system shall ensure that operational impacts incurred for collision avoidance and congestion management are considered in total and not separately [↓ RC-73]

Req-100. ATM system shall maintain a consolidated state of the airspace to ensure synchronized traffic management decision-making [↓ Resp-5]

B.2 Defining Control Elements to Meet System Requirements

The figures in this section show how the requirements defined in the previous section were used to generate the five control responsibilities and their associated control actions and feedback identified in design iteration 1. Each of these responsibilities and a simplified version of their corresponding control actions and feedback were shown on the revised conceptual architecture shown in Figure 28 in Section 4.2.4. Each control element is traced to the constraint or requirement used to generate it using the links in square braces.

Resp-1: Coordinate the movement of aircraft to prevent conflicts *[Req-4]*

RC-2: Account for planned trajectory when identifying conflicts *[Req-10]*

RC-3: Ensure that coordination provided to the aircraft is within the capabilities of the aircraft *[Req-15]*

RC-4: Ensure coordination decisions do not cause secondary conflicts *[Req-17]*

RC-6: Ensure that aircraft have received coordination, are executing it correctly and that the risk of collision is no longer present *[Req-5]*

RC-7: Account for any users' constraints on mission execution in addition to access priorities when coordinating aircraft *[Req-21]*

RC-8: Respond to impending interference and issue coordination instructions within <TBD> time *[Req-24]*

RC-15: Continue resolving conflicts even if one or more aircraft are unable to communicate or are not responding *[Req-12]*

RC-26: Ensure that aircraft have received the coordination being communicated *[Req-13]*

RC-27: If coordination was not effective, coordination is evaluated again to ensure that risks are adequately mitigated *[Req-50]*

RC-31: Ensure that alternative movement option are considered when coordinating aircraft *[Req-57]*

RC-54: Ensure that initiated traffic management plans are used to influence trajectory modifications, alternative trajectory selection, and airspace access management *[Req-61]*

RC-58: Confirm alternative trajectories are available for any proposed coordination *[Req-83]*

RC-61: Account for reasons that a trajectory modification was ineffective when selecting new trajectory modifications *[Req-85]*

RC-71: Check in with affected aircraft on preferred trajectory modification if unable to meet all operational constraints *[Req-86]*

Process Model Parts & Required Feedback/Inputs

<u>Feedback from the aircraft:</u>

- Acknowledgement of trajectory modifications *[RC-26]*
- Reason for trajectory deviation *[RC-61]*
- Preferred trajectory modification *[RC-71]*

<u>Input from Resp-2:</u> Active traffic management program *[RC-54]*

<u>Input from Resp-3:</u>

- Confirm trajectory modifications *[RC-58]*
- Alternate trajectories *[RC-31, RC-58]*

<u>Input from Resp-5:</u>

- Aircraft not communicating *[RC-15]*
- Consolidated airspace state *[RC-2, RC-3, RC-4, RC-7, RC-27]*

<u>Input from Regulators:</u> Airspace access priorities *[RC-7]*

<u>Internal process model variables:</u> Unresolved conflicts *[RC-6]*

Required Control Actions/Outputs

<u>Control actions to the aircraft:</u>

- Trajectory modifications *[Resp-1]*
- Request acknowledgement of trajectory modifications *[RC-26]*
- Trajectory modification options *[RC-71]*

<u>Output to Resp-5:</u> Trajectory modifications *[Resp-1]*

<u>Output to Resp-3:</u> Proposed trajectory modifications *[RC-58]*

Figure B-1: Defined control elements for Resp-1

Resp-2: Ensure that sufficient capacity is available to detect and coordinate all aircraft that have or will need access to the airspace *[Req-3]*

RC-21: Have reserve capacity set aside to provide additional coordination during unexpected emergencies or incidents *[Req-34]*

RC-22: Ensure that capacity determinations account for both traffic density and coverage area *[Req-45]*

RC-23: A consolidated view of the airspace should be used to manage air traffic in accordance with ATM capacity (Resp-2) and generate alternate trajectories (Resp-3) *[Req-73]*

RC-54: Ensure that initiated traffic management plans are used to influence trajectory modifications (Resp-2), alternative trajectory selection (Resp-3), and airspace access management (Resp-4) decisions *[Req-61]*

Process Model Parts & Required Feedback/Inputs

Feedback from Resp-1: Current workload (of controllers resolving conflicts) *[RC-21]*

Feedback from Resp-5: Consolidated airspace state *[RC-23]*

Internal Process Model Variables: Current & historical congestion level *[Resp-, RC-21]*

Required Control Actions/Outputs

Control action to Resp-2, Resp-3, Resp-4: Initiate traffic management program *[Resp-2, RC-54]*

Figure B-2: Defined control elements for Resp-2

Resp-3: Ensure that acceptable coordination options are always available for aircraft to avoid violation of minimum separation *[Req-6]*

RC-23: A consolidated view of the airspace should be used to manage air traffic in accordance with ATM capacity (Resp-2) and generate alternate trajectories (Resp-3) *[Req-73]*

RC-25: Ensure that there is sufficient airspace available (or low enough density) available (Resp-4) to allow alternative trajectories to be established *[Req-51]*

RC-28: Ensure that access to the airspace is managed (Resp-4) in accordance with what is needed to ensure that acceptable coordination options are always available (Resp-3) *[Req-52]*

RC-50: Recompute alternative movement options within <TBD> time so that re-evaluations can happen continuously *[Req-74]*

RC-54: Ensure that initiated traffic management plans are used to influence trajectory modifications (Resp-2), alternative trajectory selection (Resp-3), and airspace access management (Resp-4) decisions *[Req-61]*

Process Model Parts & Required Feedback/Inputs

Feedback from Resp-1: Proposed trajectory modifications *[Resp-3]*

Input from Resp-2: Initiate traffic management program *[RC-54]*

Feedback from Resp-5: Consolidated airspace state *[RC-23]*

Required Control Actions/Outputs

Control actions to Resp-1:

- Alternate trajectories *[Resp-3]*
- Confirm trajectory modifications *[Resp-3]*

Control action to Resp-4: Alternate trajectories *[RC-25]*

Figure B-3: Defined control elements for Resp-3

Resp-4: Only allow as many users to access the airspace as it is capable of detecting, tracking and coordinating *[Req-8]*

RC-28: Ensure that access to the airspace is managed (Resp-4) in accordance with what is needed to ensure that acceptable coordination options are always available (Resp-3) *[Req-52]*

RC-54: Ensure that initiated traffic management plans are used to influence trajectory modifications (Resp-2), alternative trajectory selection (Resp-3), and airspace access management (Resp-4) decisions *[Req-61]*

RC-63: Ensure that tracking and trajectory information is available for all aircraft before the aircraft is allowed to enter the UAM operating environment *[Req-88]*

RC-64: Ensure that a conflict-free trajectory is available (Resp-1) for an aircraft before allowing it to enter UAM airspace *[Req-89]*

RC-72: Propose other flight plan options (e.g., earlier departure, different arrival/departure aerodrome) if an excessive operational constraint will be incurred *[Req-97]*

Process Model Parts & Required Feedback/Inputs

<u>Feedback from the aircraft:</u>

- Flight plans *[Resp-4]*
- Preferred flight plan modifications *[RC-72]*

<u>Inputs from Resp-1:</u>

- Confirm trajectory is conflict free *[RC-64]*
- Trajectory modifications *[RC-64]*

<u>Input from Resp-2:</u> Initiate traffic management program *[RC-54]*

<u>Input from Resp-3:</u> Alternate trajectories *[RC-28]*

<u>Input from Resp-5:</u>

- Consolidated airspace state *[Resp-4]*
- Aircraft info available *[RC-63]*

Required Control Actions/Outputs

<u>Control actions to aircraft:</u>

- Flight plan modifications *[Resp-4]*
- Approve/reject access *[Resp-4]*
- Flight plan modification options *[RC-72]*

<u>Output to Resp-1:</u> Incoming aircraft *[RC-64]*

<u>Output to Resp-5:</u> Incoming aircraft *[RC-63]*

Figure B-4: Defining control elements for Resp-4

Resp-5: Maintain a consolidated state of the airspace for use in traffic management decision-making *[Req-100]*

RC-1: Track all aircraft in the airspace within <TBD> performance requirements to keep them separated *[Req-1]*

RC-19: Prevent the manipulation or tampering of data used for detecting and tracking aircraft *[Req-9]*

RC-29: Ensure that coordination decisions (Resp-1) account for whether tracking information is missing/out-of-date (Resp-5) *[Req-5]*

RC-32: Ensure that information about the intent, mission, acceptable operational impacts and future intended movements of aircraft is available, does not contain errors and is kept up to date *[Req-11]*

RC-62: Ensure that track and trajectory information is available for all aircraft within <TBD distance> of the UAM operating environment *[Req-87]*

RC-74: Verify erroneous detections within <TBD time> before choosing to ignore them *[Req-2]*

RC-75: Detect when information needed to identify and track aircraft is missing, delayed, erroneous or not available and take action to restore that information *[Req-7]*

Process Model Parts & Required Feedback/Inputs

Feedback from the aircraft:

- Aircraft Track *[Resp-5, RC-1, RC-62]*
- Planned trajectory *[Resp-5, RC-62]*
- Mission & operational constraints *[Resp-5, RC-32]*
- Aircraft navigational capabilities *[Resp-5, RC-32]*

Feedback from Resp-4: Incoming aircraft to UAM airspace *[RC-62]*

Internal Process Model Variable: Aircraft authenticity *[RC-19]*

Required Control Actions/Outputs

Control actions to the aircraft:

- Request aircraft track *[RC-75]*
- Request planned trajectory *[RC-75]*
- Reject aircraft track/trajectory *[RC-19, RC-74]*

Control actions to Resp-1:

- Consolidated airspace state *[Resp-5]*
- Aircraft not communicating *[RC-29]*

Figure B-5: Defining control elements for Resp-5

Appendix C Design Iteration 1 – STPA Analysis of Initial Conceptual Architecture

As described in Section 4.2.3, the initial conceptual architecture that was created in design iteration 1 (Figure 26) was analyzed by updating the initial STPA analysis shown in Section 4.1 and Appendix A. Since the focus of this research is on the collision avoidance aspect of ATM, this updated STPA analysis focused on analyzing the *Trajectory Modifications* control action. This section shows the results of this updated SPTA analysis.

C.1 Unsafe Control Actions (UCAs) for *Coordination* Control Action

Since this is an update of the initial STPA analysis, the system-level losses and hazards are the same as those shown in Table 7 and Table 8. Table C-1 and Table C-2 shows the refined UCAs associated with the *Trajectory Modifications* control action. For each refined UCA, a link to the corresponding abstract version of the UCA shown in Table A-1 is included in square braces along with the links to the system hazards. Any UCAs in Table A-1 that do not have refined UCAs associated with the *Trajectory Modifications* control action are not shown in Table C-1 and Table C-2.

Table C-1: Refined UCAs for Trajectory Modifications Control Action

Not Providing	Providing
UCA-1.1.1: Trajectory modifications are not provided when the trajectories of two aircraft are in conflict *[H-1, H-3] [UCA-1.1]*	**UCA-1.16.1:** Trajectory modifications are provided when those modified trajectories interfere with approach/arrival course/trajectory for a nearby airport *[H-1, H-3, H-6] [UCA-1.16]*
UCA-1.2.1: Trajectory modifications are not provided when the trajectory needed by an aircraft experiencing an emergency conflicts with other aircraft *[H-1, H-2, H-3] [UCA-1.2]*	**UCA-1.18.1:** Trajectory modifications are provided that will result in excessive environmental effect *[H-4] [UCA-1.18]*
UCA-1.3.1: Trajectory modifications are not provided when the arrival trajectory of a UAM aircraft at a conventional airport will conflict with the approach course used by conventional aviation aircraft *[H-1, H-3] [UCA-1.3]*	**UCA-1.19.1:** Trajectory modifications are provided that interfere with the operations of other NAS users *[H-3, H-6] [UCA-1.19]*
UCA-1.5.1: Trajectory modifications are not provided when UAM aircraft are about to fly into a section of airspace where air traffic must be excluded (e.g., for safety or security reasons) *[H-5] [UCA-1.5]*	**UCA-1.20.1:** Trajectory modifications are provided that UAM aircraft are not fully capable of executing *[H-1, H-2, H-3, H-6] [UCA-1.20]*
UCA-1.8.1: Trajectory modifications are not provided when UAM aircraft interfere with the flight of an emergency response aircraft *[H-1, H-3] [UCA-1.8]*	**UCA-1.21.1:** Trajectory modifications are provided that causes a collision with an obstacle or terrain *[H-1, H-2, H-3, H-5] [UAC-1.21]*

Not Providing	Providing
UCA-1.8.2: Trajectory modifications are not provided when a higher priority flight incurs an unacceptable operational impact (e.g., delay) due to UAM flights that are occurring *[H-1, H-3] [UCA-1.8]*	**UCA-1.22.1:** Trajectory modifications are provided that causes a collision with another aircraft *[H-1, H-2, H-3] [UCA-1.22]*
UCA-1.9.1: Trajectory modifications are not provided when UAM aircraft operations have excessive noise, privacy or emissions impacts *[H-4] [UCA-1.9]*	**UCA-1.24.1:** Trajectory modifications are provided when those trajectories allocate more airspace than necessary to prevent collisions *[H-3, H-6] [UCA-1.24]*
UCA-1.10.1: Trajectory modifications are not provided when UAM aircraft need to be sequenced for arrival to a conventional airport *[H-3] [UCA-1.10]*	**UCA-1.24.2:** Trajectory modifications are provided that exceed the operational constraints for the aircraft needed to execute the trajectory *[H-3, H-6] [UCA-1.24]*
UCA-1.11.1: Trajectory modifications are not provided when UAM aircraft have overlapping arrival or departure trajectories *[H-3] [UCA-1.11]*	**UCA-1.24.3:** Trajectory modifications are provided when the trajectory of an aircraft is already valid and optimal *[H-3, H-6] [UCA-1.24]*
UCA-1.12.1: Trajectory modifications are not provided when the trajectory of an aircraft conflicts with an obstacle or terrain *[H-1] [UCA-1.12]*	**UCA-1.25.1:** Trajectory modifications are provided that causes the aircraft to traverse adverse weather that it is not equipped to handle *[H-1, H-2, H-3, H-5] [UCA-1.25]*
UCA-1.13.1: Trajectory modifications are not provided when the trajectory of a UAM aircraft will take it toward inclement weather that exceeds the capabilities of the aircraft *[H-1, H-2, H-3] [UCA-1.13]*	**UCA-1.27.1:** Trajectory modifications are provided that forces a UAM aircraft to use airspace where the ride is unpleasant or unsafe for passengers *[H-2, H-6] [UCA-1.27]*
	UCA-1.28.1: Trajectory modifications are provided when they do not satisfy the priority needs of the aircraft *[H-1, H-2, H-3] [UCA-1.28]*

Table C-2: Too early/late and applied too long/stopped too soon UCAs for Trajectory Modifications Control Action

Too Early / Too Late	Applied Too Long / Stopped Too Soon
UCA-1.29.1: Trajectory modifications are provided too late after the trajectories of two aircraft are in conflict *[H-1, H-2, H-3] [UCA-1.29]*	**UCA-1.35.1:** Trajectory modifications are provided for too long when conditions have changed such that the original trajectory modifications are no longer valid *[H-1, H-2, H-3, H-4, H-6] [UCA-1.35]*
UCA-1.30.1: Trajectory modifications are provided too late after an aircraft requires an immediate change in trajectory (e.g., to address an emergency) *[H-1, H-2, H-3] [UCA-1.30]*	**UCA-1.36.1:** Trajectory modifications stop being provided too soon before the emergency experienced by UAM aircraft is resolved *[H-1, H-2, H-3] [UCA-1.36]*
UCA-1.32.1: Trajectory modifications are provided too late after environmental effects of UAM operations have exceeded acceptable levels *[H-4] [UCA-1.32]*	**UCA-1.37.1:** Trajectory modifications stop being provided too soon to prevent UAM aircraft from entering a restricted section of airspace when air traffic still needs to be excluded from that section of airspace *[H-2, H-4, H-6] [UCA-1.37]*
UCA-1.33.1: Trajectory modifications are provided too late after an aircraft enters a volume of airspace where congestion has already exceeded acceptable levels *[H-1, H-3, H-4, H-6] [UCA-1.33]*	**UCA-1.38.1:** Trajectory modifications stop being provided too soon before environmental effects of system operation have returned to acceptable levels *[H-4] [UCA-1.38]*
UCA-1.34.1: Trajectory modifications are provided too late after UAM aircraft have already interfered with the operations of another airspace user *[H-3, H-6] [UCA-1.34]*	**UCA-1.39.1:** Trajectory modifications are provided for too long after environmental effects of system operation have returned to acceptable levels *[H-3] [UCA-1.39]*

C.2 Causal Scenarios for Selected UCAs

This section shows the causal scenarios that were identified for select UCAs highlighted in blue in Table C-1 and Table C-2. Two types of links are included at the end of each requirement. First, if additional system requirements were generated to mitigate or prevent that scenario, the requirement is linked in square braces. These requirements are ultimately used to make modifications to the conceptual architecture.

Second, if the scenario was used in the structural design process to generate an assignment constraint, that assignment constraint is indicated in curly braces and blue font.

Scenarios for UCA-1.1.1: Trajectory modifications are not provided when the trajectories of two aircraft are in conflict *[H-1, H-3]*

CS-1.1.1-1. Feedback of the potential conflict is received but trajectory modifications are not provided because:

CS-1.1.1-1.1. Resp-3 does not confirm that the trajectory modifications still have alternate trajectory options. As a result, Resp-1 is unable to issue trajectory modifications to resolve the collision *{Resp-1 = Resp-3}*

CS-1.1.1-1.2. Resp-1 is pre-occupied with resolving one set of conflicts and therefore does not attend to the feedback about another imminent collision. As a result, Resp-1 does not issue trajectory modifications to resolve the imminent collision *{(Resp-1=Aircraft) ∨ (Resp-1=Aircraft ∧ ATM)}*

CS-1.1.1-1.3. Resp-1 is unable to determine possible movement options due to a component failure that prevents trajectory modifications from being computed. As a result, no trajectory modifications are issued *{(Resp-1=Aircraft) ∨ (Resp-1=Aircraft ∧ ATM)}*

CS-1.1.1-1.4. It takes so long to identify a conflict-free solution that no trajectory modification is issued before the collision occurs. This could occur if, for example, the airspace is so densely utilized that resolving a limited initial conflict requires changes to a large number of aircraft trajectories to accommodate the initial trajectory modifications required. *{(Resp-1=Aircraft) ∨ (Resp-1=Aircraft ∧ ATM)}*

CS-1.1.1-1.5. Resp-1 is unable to select possible movement options because there is no combination of alternative trajectories that will prevent all conflicts. For example, the emergency trajectory of one aircraft is such that there is no set of suitable trajectories for all other aircraft that can be selected that are conflict free

CS-1.1.1-1.6. Resp-5 wrongly believes the feedback it receives about the track of an aircraft is erroneous and omits it from the set of verified tracks. As a result, Resp-1 does not recognize the collision because the aircraft track needed to recognize it was omitted

CS-1.1.1-1.7. Resp-1 is unable to provide trajectory modifications to the aircraft because Resp-3 rejects its proposed trajectory modifications. This could occur if Resp-1 and Resp-3 do not synchronously process inputs from Resp-5 to remove an aircraft track and thus have inconsistent process models of the state of the airspace. As a result, they cannot agree on a set of trajectory modifications that have alternate trajectories available *[Req-100]*

CS-1.1.1-2. Feedback is not received of the potential conflict because:

CS-1.1.1-2.1. At least one of the two aircraft enters the UAM environment without its tracking and trajectory information having been fully received. This could occur for several reasons: (1) The aircraft is inadequately equipped and cannot be tracked using normal means, (2) the aircraft is non-cooperative (e.g., malicious aircraft) or (3) the aircraft enters the UAM environment before tracking and trajectory information can be fully collected. As a result, Resp-1 either does not know the aircraft is there or has the wrong belief about the trajectory of that aircraft and therefore wrongly believes that no collision is imminent *[Req-87, Req-88]* *{Resp-1 = ATM}*

CS-1.1.1-2.2. Resp-1 receives either inaccurate feedback about the equippage/technical specifications of an aircraft or out-of-date feedback about the level of precision with which the aircraft can execute a trajectory (e.g., GPS outage/blockage/degradation). As a result, it wrongly believes they are capable of more precise navigation than they actually are and therefore does not believe two aircraft are on collision trajectories even though they are *{(Resp-1=Aircraft) ∨ (Resp-1=Aircraft ∧ ATM)}*

CS-1.1.1-2.3. This could occur if it does not receive timely feedback on the presence of new ground hazards (e.g., a new construction crane). As a result, it does not believe a collision is imminent and does not try to modify aircraft trajectories to avoid the collision *{(Resp-1=Aircraft) ∨ (Resp-1=Aircraft ∧ ATM)}*

CS-1.1.1-2.4. When Resp-1 checked the trajectories of all aircraft, their trajectories were not in conflict. However, while resolving another conflict, the trajectories of these aircraft do become in conflict and this is not noticed/resolved until after the previous set of conflicts are resolved. If it takes long enough for the prior set of conflicts to be resolved, the system may not be able to identify and adequately resolve the conflict before a collision occurs *{(Resp-1=Aircraft) ∨ (Resp-1=Aircraft ∧ ATM)}*

CS-1.1.1-3. Trajectory modifications are provided to resolve the conflict, but they are not received by the aircraft because:

CS-1.1.1-3.1. An equipment failure or malicious interference prevents the trajectory modification from being issued to the aircraft. As a result, the aircraft continues on the old trajectory, not realizing that a trajectory modification has been provided *{(Resp-1=Aircraft) ∨ (Resp-1=Aircraft ∧ ATM)}*

CS-1.1.1-4. Trajectory modifications are provided to resolve the conflict and they are received by the aircraft. However, the conflict is not resolved and a collision occurs. This could occur because:

CS-1.1.1-4.1. One of the aircraft receives the trajectory modification but does not execute the trajectory modification because they wrongly believe that the provided trajectory modification would result in another violation of minimum separation. This might occur because aircraft are only provided with their trajectory modification and have no awareness of trajectory modifications provided to other aircraft. As a result, they wrongly believe that they are on a collision course with another aircraft even though they are not because that aircraft is also about to change trajectories. They therefore ignore the provided trajectory modification and make an independent decision which ultimately violates minimum separation *{(Resp-1=Aircraft) ∨ (Resp-1=Aircraft ∧ ATM)}*

CS-1.1.1-4.2. Trajectory modifications are provided and received by the aircraft but the conflict is not resolved. This could occur if the aircraft does not execute the trajectory to the level of precision expected by Resp-1 when it generated the trajectory modifications. This could occur either because the aircraft lacks the required equippage, the required equipment has failed or the equipment is momentarily degraded by environmental conditions (e.g., temporary GPS outage, wind gusts, poor weather etc). As a result, the provided trajectories do not adequately resolve the conflict *[Req-85, Req-90, Req-91]*

Scenarios for UCA-1.8.1: Trajectory modifications are not provided when UAM aircraft interfere with the flight of an emergency response aircraft *[H-1, H-3]*

CS-1.8.1-1. Feedback indicating that a UAM aircraft will interfere with the flight of an emergency response aircraft is received. However, trajectory modifications are not provided because:

CS-1.8.1-1.1. If the airspace is densely occupied, Resp-1 may not be able to find a solution to clear a path for the emergency response aircraft. Alternatively, it might take so long to resolve all the conflicts such that trajectory modifications are not issued before the emergency response aircraft needs to depart. *{(Resp-1=Aircraft) ∨ (Resp-1=Aircraft ∧ ATM)}*

CS-1.8.1-1.2. That feedback is only received at the last minute due to the emergency response aircraft modifying their trajectories quickly in response to an evolving emergency event (e.g., a bad accident, a wild fire). If Resp-1 is unable to respond and modify the trajectories of UAM aircraft in response, they can end up interfering with the emergency response flight. *{(Resp-1=Aircraft) ∨ (Resp-1=Aircraft ∧ ATM)}*

CS-1.8.1-2. Feedback indicating that a UAM aircraft will interfere with the flight of an emergency response aircraft is not received because:

CS-1.8.1-2.1. Interference is only being monitored with respect to trajectory and not track. Thus, if either aircraft is not following its planned trajectory precisely (e.g., either slightly delayed or slightly ahead), the UAM aircraft could interfere with the emergency response flight even though that interference is not reflected in their planned trajectories. *[Req-90, Req-91]*

CS-1.8.1-4. Feedback indicating that a UAM aircraft will interfere with the flight of an emergency response aircraft is received and appropriate trajectory modifications are provided. However, that interference still occurs because:

CS-1.8.1-4.1. The trajectory modification does not include an expectation of immediate action and therefore the UAM aircraft may not execute the modified trajectory as quickly as necessary. *[Req-94]*

Scenarios for UCA-1.12.1: Trajectory modifications are not provided when the trajectory of an aircraft conflicts with an obstacle or terrain *[H-1]*

CS-1.12.1-1. Feedback indicating that the trajectory of an aircraft conflicts with an obstacle or terrain is received but trajectory modifications are not provided. This could occur if:

CS-1.12.1-1.1. The capabilities of the aircraft are compromised (e.g., stronger winds than the aircraft can handle, partial system failure) and there are no suitable options available that will sufficiently avoid the obstacle and be within the compromised capabilities of the aircraft.

CS-1.12.1-2. Feedback indicating that the trajectory of an aircraft conflicts with an obstacle or terrain is not received because:

CS-1.12.1-2.1. If the obstacle is new and/or temporary (e.g., a tall crane or a temporary structure near a UAM aerodrome) and the presence of this obstacle has not been disseminated to ATM. Furthermore, such an obstacle may not be easily detected by the aircraft themselves if obscurants are present *[Req-96]* *{(Resp-1=Aircraft) ∨ (Resp-1=Aircraft ∧ ATM)}*

<u>**Scenarios for UCA-1.22.1:**</u> Trajectory modifications are provided that causes a collision with another aircraft *[H-1, H-2, H-3]*

CS-1.22.1-1. Feedback indicating that the trajectories of two aircraft are in conflict was received. However, those trajectory modifications are provided anyway because:

CS-1.22.1-1.1. An emergency or last-minute airspace restriction occurs, requiring large numbers of aircraft to modify their trajectories. As a result, the system is forced to quickly modify trajectories for numerous aircraft before it can adequately consider the collision implications of the new trajectories. The system therefore selects the trajectory modifications that result in the fewest collisions and issues those even though some trajectories have collisions. *[Req-101, Req-102]* *{Resp-1 = ATM}*

CS-1.22.1-1.2. Resp-1 believes that it can issue further trajectory modifications later to prevent collision. This could occur if, for example, the system is attempting to prevent a more urgent collision (e.g., an emergency) and believes it can make a faster decision by issuing a trajectory modification that prevents the urgent collision even if it subsequently causes a later collision. However, Resp-1 does not return to correct that collision (e.g., because it becomes preoccupied resolving other collisions) and thus the collision occurs *[Req-101, Req-102]* *{(Resp-1=Aircraft) ∨ (Resp-1=Aircraft ∧ ATM)}*

CS-1.22.1-1.3. Aircraft trajectories change while the system is resolving a set of conflicts. During the conflict resolution process, Resp-1 does not update its process model of the trajectories of aircraft even though those might change. As a result, while the trajectory modifications are being generated, Resp-1 selects trajectory modifications that were not in conflict during the selection process but are in conflict based on the most current set of aircraft trajectories. *{(Resp-1=Aircraft) ∨ (Resp-1=Aircraft ∧ ATM)}*

CS-1.22.1-2. Feedback did not indicate that the provided trajectory modifications will cause a collision with another aircraft because:

CS-1.22.1-2.1. Resp-1 does not receive timely feedback of ground hazards (e.g., a new construction crane being erected) and believes that the trajectory modification it is providing will not cause a conflict with that ground hazard. As a result, it issues that trajectory modification, unaware that it will cause a collision *{(Resp-1=Aircraft) ∨ (Resp-1=Aircraft ∧ ATM)}*

CS-1.22.1-2.2. Resp-1 does not receive timely information about the aircraft capabilities or aircraft type. For example, it could wrongly believe the aircraft is capable of more precise navigation than it actually is. As a result, the system approves trajectory modifications that it wrongly believes do not contain collisions but a collision does actually occur. *{(Resp-1=Aircraft) ∨ (Resp-1=Aircraft ∧ ATM)}*

CS-1.22.1-3. Feedback indicating that the trajectories of two aircraft are in conflict was received and trajectory modifications are provided that do not result in a collision. However, the aircraft still receive trajectory modifications that result in a collision because:

CS-1.22.1-3.1. During transmission of appropriate trajectory modifications to the aircraft, part of the trajectory modification is dropped (e.g., due to a partial/temporary communications failure). As a result, the aircraft only receives part of the trajectory modifications and the part that is received by the aircraft is in collision with another aircraft trajectory *{(Resp-1=Aircraft) ∨ (Resp-1=Aircraft ∧ ATM)}*

CS-1.22.1-3.2. The trajectory modifications are transmitted without accompanying navigation accuracy parameters (e.g., lateral or vertical tolerances). As a result, the trajectory modifications are not effective in preventing a collision because they contain insufficient information for the aircraft to carry them out as intended. *[Req-94]*

CS-1.22.1-4. Feedback indicating that the trajectories of two aircraft are in conflict was received and trajectory modifications that do not result in a collision are provided and received by the aircraft. However, a collision still occurs because:

CS-1.22.1-4.1. The aircraft does not carry them out as intended (e.g., not within navigational tolerances) such that a collision does actually occur. *[Req-85, Req-90, Req-91]*

<u>**Scenarios for UCA-1.24.1:**</u> Trajectory modifications are provided when those trajectories allocate more airspace than necessary to prevent collisions *[H-3, H-6]*

CS-1.24.1-1. Feedback indicates that trajectory modifications will allocate more airspace than necessary to prevent collisions is received. However, those trajectory modifications are still provided because:

CS-1.24.1-1.1. As part of its decision-making to issue trajectory modifications, Resp-1 considers whether a collision remains unresolved. If Resp-1 wrongly believes that a collision remains unresolved because the aircraft is not adequately following the trajectory, it may responds by issuing further trajectory modifications that unnecessarily expands the amount of airspace reserved for that trajectory. As a result, more airspace is used to serve that flight than is necessary. *[Req-84]*

CS-1.24.1-1.2. Resp-1 believes that a weather or other event will occur soon will compromise the ability of aircraft to follow more precise trajectory or the ability to track them precisely. As a result, the system issues these expanded trajectory modifications anyway to protect airspace safety even though they consume more airspace than necessary at the current time. In addition, if the anticipated event does not ultimately occur, these expanded trajectory modifications will not have been necessary at all.

CS-1.24.1-1.3. Resp-1 wrongly believes that there is no better option that allocates less airspace. This could occur if Resp-3 does not remove a track for a non-existent aircraft, but Resp-1 does. As a result, Resp-3 generates alternate trajectories for the non-existent aircraft and passes those alternate trajectories to Resp-1. Thus, Resp-1 wrongly believes it needs to avoid routing aircraft through the airspace allocated for the alternate trajectory of the non-existent aircraft even though doing so is unnecessary. *[Req-100]*

CS-1.24.1-1.4. Resp-1 receives feedback that the aircraft is not following its trajectory exactly as expected but does not receive enough feedback to know exactly how much extra space buffer to provide. For example, this could occur if the aircraft is having trouble tracking its trajectory accurately due to wind but Resp-1 does not receive feedback about the extent to which the aircraft can hold trajectory. As a result, Resp-1 provides trajectory modifications to allow extra room even though they might be unnecessary *{(Resp-1=Aircraft) ∨ (Resp-1=Aircraft ∧ ATM)}*

CS-1.24.1-2. Feedback indicating trajectory modifications allocate more airspace than necessary to prevent a collision is not received because:

CS-1.24.1-2.1. Resp-1 either does not receive feedback about the demand for flights or receives that feedback with a delay. As a result, the system wrongly believes that demand for flights is lower than it really is and it therefore generates trajectories that provide more spacing between flights, wrongly believing that the unused airspace can be used to increase separation between flights and thus increase airspace safety. *{Resp-1 = ATM}*

CS-1.24.1-2.2. Resp-3 generates an initial set of alternate trajectories based on the current trajectories of the aircraft. However, those trajectories change such that more efficient alternate trajectories become available but Resp-3 does not re-evaluate its selected alternate trajectories. As a result, Resp-1 selects trajectory modifications based on that inefficient set of alternate trajectories selected by Resp-3. *{Resp-1 = Resp-3}*

CS-1.24.1-2.3. The aircraft try to reserve more airspace than necessary for themselves as a safety margin and therefore indicate to Resp-1 that they are capable of less precise navigation than they actually are. As a result, Resp-1 bases its selection of trajectory modifications based on that feedback, not realizing that those trajectories consume more airspace than necessary. *{Resp-1 = ATM}*

Scenarios for UCA-1.24.2: Trajectory modifications are provided that exceed the operational constraints for the aircraft needed to execute the trajectory *[H-3, H-6]*

CS-1.24.2-1. Feedback indicates that trajectory modifications will exceed the operational constraints of an aircraft is received. However, those trajectory modifications are still provided because:

CS-1.24.2-1.1. Although the system recognizes that operational constraints would be exceeded (e.g., a significant extension of the flight path), it believes that it would be optimal to minimize the operational impacts to other aircraft (that might have fly at higher speeds or be carrying more people) and as a result saddles a single aircraft with numerous operational impacts (e.g., multiple flight path extensions or delays) *{Resp-1 = ATM}*

CS-1.24.2-1.2. Although Resp-1 recognizes that operational constraints would be exceeded, it is unable to meet all operational constraints and airspace constraints. This is especially likely to occur if one or more aircraft experience an emergency requiring unexpected and immediate route changes. In such a condition, Resp-1 chooses to meet the airspace constraints (e.g., to give an aircraft experiencing an emergency priority) and violate the operational constraints, thus issuing trajectory modifications that violate operational constraints for an aircraft

CS-1.24.2-1.3. Air traffic circumstances are such that there were no available options that would meet all operational constraints. Although Resp-1 begins the process of negotiating with aircraft on their preferred trajectory modifications, it decides it needs to take action before the negotiation process is complete (e.g., collision is imminent). As a result, the system issues trajectory modifications that are not aligned with the aircraft's constraint priorities. *[Req-86]*

CS-1.24.2-2. Feedback indicating that trajectory modifications will exceed the operational constraints of an aircraft is not received because:

CS-1.24.2-2.1. Some operational constraints change over time (e.g., diversion options become more restricted due to fuel remaining as a flight progresses). If the system does not receive timely feedback indicating a change to the operational constraints when it selects alternative trajectories, it could select trajectory modifications based on out-of-date or incomplete operational constraints, not realizing that the constraints have changed or new ones now exist.

CS-1.24.2-3. Feedback indicates that trajectory modifications will exceed the operational constraints of an aircraft is received and appropriate trajectory modifications are selected and received by the aircraft. However, the modified trajectories still exceed operational constraints. This could occur if:

CS-1.24.2-4. The trajectory modifications did not exceed operational constraints when they were issued or initially carried out but it is later realized that they do. For example, an aircraft may believe that they have enough fuel to accept a trajectory modification but later realize they do not. Another example might be a medical flight that initially believes it can accept a flight delay only for the patient's condition to worsen or be poor enough that the delay was actually or becomes unacceptable. *{(Resp-1=Aircraft) ∨ (Resp-1=Aircraft ∧ ATM)} [Req-95]*

<u>Scenarios for UCA-1.24.3:</u> Trajectory modifications are provided when the trajectory of an aircraft is already valid and optimal *[H-3, H-6]*

CS-1.24.3-1. Feedback indicates that the trajectory of an aircraft is already valid and optimal but trajectory modifications are still provided because:

- **CS-1.24.3-1.1.** Resp-1 is told to modify trajectories as part of a traffic management program being activated to expand capacity. As such, although the trajectory is already valid and optimal for that aircraft, the system modifies the trajectory in a way that is less optimal (but still collision free) to implement the traffic management program
- **CS-1.24.3-1.2.** Resp-1 knows that the trajectory of an aircraft is already valid and optimal, it is forced to make a trajectory modification for higher-priority traffic (e.g., an emergency responder flight or an aircraft experiencing an emergency). As a result, it makes a trajectory modification that is no longer optimal and that results in a delay or even the aircraft being unable to complete its mission entirely
- **CS-1.24.3-1.3.** Resp-2 believes that a traffic surge event is about to occur (e.g., UAM operators initiate a series of flights in response to ride requests after a sporting event) and initiates a traffic management program to manage capacity in anticipation of the surge. As part of implementing that traffic management program, the trajectory of aircraft are modified to comply with the traffic management program. However, if that surge event never occurs (e.g., those ride requests never get fulfilled because riders give up or cancel rides), those trajectory modifications will have been unnecessary *[Req-30, Req-31]*

CS-1.24.3-2. Feedback does not indicate that the trajectory of an aircraft is already valid and optimal. This could occur if:

- **CS-1.24.3-2.1.** Resp-1 receives erroneous track data for a flight due to degradation or a flaw in how aircraft track information is generated (e.g., bad weather). As a result, the system modifies the trajectories of other flights unnecessarily to avoid a collision with this aircraft
- **CS-1.24.3-2.2.** Resp-1 does not receive feedback about the trajectory of an aircraft (e.g., an inadequately equipped aircraft that did not communicate intentions beforehand) and attempts to infer the trajectory of the aircraft from its track. As a result, it has an incorrect belief of the trajectory of the aircraft and wrongly believes that a collision is imminent. As a result, the trajectories of aircraft are unnecessarily altered to resolve a conflict that was not present *[Req-87, Req-88]*

CS-1.24.3-4. Feedback indicates that the trajectory of an aircraft is already valid and optimal and no trajectory modifications are provided. However, the aircraft's trajectory is modified anyway. This could occur if:

- **CS-1.24.3-4.1.** The aircraft deviates from a valid and optimal trajectory due to either deliberate actions or unintended consequences. Deliberate actions include modifying trajectory to avoid a temporary obstacle or hazard that was not previously known (e.g., a crane temporarily placed on a building). Unintended consequences could be sudden wind gusts or cloud buildup that the aircraft is unable to counteract or needs to maneuver around. *[Req-85, Req-90, Req-91]*

<u>**Scenarios for UCA-1.29.1:**</u> Trajectory modifications are provided too late after the trajectories of two aircraft are in conflict *[H-1, H-2, H-3]*

CS-1.29.1-1. Feedback indicating that the trajectories of two aircraft are in conflict was received on time. However, trajectory modifications are provided too late. This could occur because:

CS-1.29.1-1.1. It takes Resp-1 too long to identify a solution that resolves all conflicts. As a result, trajectory modifications are issued too late to adequately resolve the collision *{(Resp-1=Aircraft) ∨ (Resp-1=Aircraft ∧ ATM)}*

CS-1.29.1-1.2. The process of generating a resolution repeatedly gets interrupted by new requests/conflicts due to the density of air traffic. As such, before the trajectory modifications can be issued, they need to be recalculated and thus the trajectory of aircraft is not modified until it is too late to enact the new trajectories to avoid a collision *{Resp-1=ATM}*

CS-1.29.1-1.3. The process of generating possible trajectory modifications and relaying them to Resp-3 is cumbersome enough (especially when traffic density is high) that by the time the trajectory modifications have been confirmed, they are issued too late to adequately prevent the collision *{Resp-1=Resp-3}*

CS-1.29.1-2. Feedback that the trajectories of two aircraft are in conflict is not received on time because:

CS-1.29.1-2.1. A degradation in the aircraft's ability to continue its mission and trajectory occurs (e.g., degraded GPS accuracy, compromised flight controls). If the aircraft does not report this degradation in a timely manner or the degradation is temporary, by the time the degradation is reported or the degradation is resolved, it is so close to the imminent collision that trajectory modifications cannot be issued sufficiently quickly to resolve the conflict *{(Resp-1=Aircraft) ∨ (Resp-1=Aircraft ∧ ATM)}*

CS-1.29.1-2.2. An aircraft is entering the UAM environment while airborne but outside it. As such, by the time tracking and trajectory information is received and the aircraft is allowed to enter the UAM environment, there is insufficient time to resolve the conflict before a collision occurs *[Req-89]*

CS-1.29.1-3. Feedback indicating that the trajectories of two aircraft are in conflict was received on time and trajectory modifications were provided on time. However, those trajectory modifications are received by the aircraft too late because:

CS-1.29.1-3.1. There is a delay in transmitting them to the aircraft. This could occur if trajectory modifications are transmitted in sets to aircraft instead of all at once. Alternatively, one of the aircraft receiving trajectory modifications is using a different communications method that is slower or not typically used (e.g., voice-based comms instead of digital text-based comms). As a result, at least one of the aircraft receive their trajectory modification too late for it to be effective at preventing the collision *[Req-92]*

CS-1.29.1-4. Feedback indicating that the trajectories of two aircraft are in conflict was received on time and trajectory modifications were provided and are received by the aircraft on time. However, the aircraft execute those trajectory modifications too late because:

CS-1.29.1-4.1. The aircraft were not expecting to receive trajectory modifications (e.g., in a critical phase of flight) and therefore is delayed more than expected in executing the new trajectory. As a result of this delay, there is not enough time to modify the trajectory of the aircraft sufficiently to prevent the collision *{(Resp-1=Aircraft) ∨ (Resp-1=Aircraft ∧ ATM)}*

Appendix D Design Iteration 1 – Analysis and Comparison of Architecture Options

This appendix shows the results from the comparison of the centralized (A_1) and decentralized (A_2) collision avoidance architecture options performed in design iteration 1. Table D-1 shows the full set of evaluation criteria that were identified and the comparison results (i.e., benefit or tradeoff) for each architecture option. To make it easier to read, the evaluation criteria in this table are sorted by type (e.g., decision making, control path). For each evaluation criterion, links are also provided in square braces to the scenario(s) in Table D-2 used to generate them.

Table D-2 then presents the full architecture comparison table that was used to generate the comparison results shown in Table D-1. Table D-2 contains (1) the scenarios used to compare the two architecture options, (2) the decisions about whether each scenario occurs for each architecture option, (3) any assumptions used to decide that a scenario does not occur for an architecture option, and (4) the evaluation criterion generated from that scenario. Note that Table D-2 only includes scenarios where behavioral differences were observed are included and scenarios where unsafe behavior was observed for both architecture options are omitted.

Table D-1: Full set of evaluation criteria for comparison of architecture options A_1 and A_2

ID	**Evaluation Criteria**	**Benefit (+) or Tradeoff (-)**	
		A1	**A2**
	Decision Making Evaluation Criteria		
EC-1	**Frequency and complexity** of trajectory modifications decisions to prevent loss of separation when <u>resolving a(n) (urgent) conflict</u> *[Scenarios 2, 14]*	⊖	⊕
EC-2	**Responsiveness** of trajectory modifications decisions to prevent inability to complete missions when <u>a high-priority flight either changes its planned trajectory or is no longer being performed</u> *[Scenario 22, 30]*	⊖	⊕
EC-3	**Responsiveness** of trajectory modifications decisions to prevent inability to complete missions when <u>providing more spacing between aircraft due to degraded navigational capabilities</u> *[Scenario 19]*	⊖	⊕
EC-4	**Ability to** make appropriate trajectory modifications to prevent loss of separation when <u>multiple conflicts occur</u> *[Scenario 5]*	⊖	⊕
EC-5	**Responsiveness** of trajectory modification decisions to prevent loss of separation when <u>resolving a multi-aircraft conflict in densely populated airspace</u> *[Scenario 4]*	⊕	⊖
EC-6	**Responsiveness** of trajectory modifications decisions to prevent loss of separation when <u>the state of the airspace changes rapidly or a conflict involves restrictive operational constraints</u> *[Scenarios 9, 29]*	⊕	⊖
EC-7	**Responsiveness** of trajectory modifications decisions to enable aircraft to complete missions when <u>reducing spacing between aircraft to accommodate additional air traffic or preventing unnecessary increases in spacing for additional safety margin</u> *[Scenarios 20, 21]*	⊕	⊖

ID	Evaluation Criteria	Benefit (+) or Tradeoff (-)	
		A1	A2
EC-8	**Responsiveness** of trajectory modification decisions to prevent inability to complete missions when <u>a high-priority flight needs to be given precedence for mission completion</u> *[Scenarios 25, 26]*	⊕	⊖
	Process Model Evaluation Criteria		
EC-9	**Situational awareness** of trajectory modifications rationale to prevent loss of separation when <u>receiving trajectory modifications to execute</u> *[Scenario 8]*	⊖	⊕
EC-10	**Level of situational awareness** of airspace state available to prevent loss of separation when <u>trajectory modifications must be identified under challenging or extremely limiting trajectory constraints</u> *[Scenario 13]*	⊕	⊖
EC-11	**Level of situational awareness** of operational impacts occurred by each flight to prevent inability to complete missions when <u>distributing operational impacts over numerous flights</u> *[Scenario 23]*	⊕	⊖
	Feedback / External Inputs Evaluation Criteria		
EC-12	**Timeliness** of ground hazards feedback to prevent loss of separation when <u>resolving a conflict</u> *[Scenarios 6, 15, 27]*	⊖	⊕
EC-13	**Timeliness** of operational constraints feedback to prevent loss of separation when <u>operational constraints are changing frequently</u> *[Scenario 24]*	⊖	⊕
EC-14	**Timeliness of** aircraft capabilities, flight conditions and operational constraints feedback to prevent loss of separation when <u>resolving a conflict</u> *[Scenarios 11, 16, 18]*	⊖	⊕
EC-15	**Use of** "confirmation of trajectory modifications" input to prevent loss of separation when <u>resolving a conflict</u> *[Scenarios 1, 10]*	⊕	⊖
EC-16	**Use of** "mutual agreement" input to prevent loss of separation when <u>resolving a conflict involving numerous aircraft and/or densely populated airspace</u> *[Scenario 28]*	⊕	⊖
	Control Path Evaluation Criteria		
EC-17	**Vulnerability** of providing trajectory modifications to prevent loss of separation when <u>a component failure compromises decision making</u> *[Scenario 3]*	⊖	⊕
EC-18	**Vulnerability** of providing trajectory modifications to prevent loss of separation when <u>errors with the communications path occurs</u> *[Scenario 17]*	⊖	⊕
EC-19	**Responsiveness** of execution of trajectory modifications to prevent loss of separation when <u>trajectory modifications have been issued</u> *[Scenarios 7, 12]*	⊖	⊕

Table D-2: Comparison results for the centralized (A_1) and decentralized (A_2) collision avoidance architecture options

ID	Scenario	Scenario Occurs?	Evaluation Criteria
1	The imminent collision is recognized and <controller(s) performing Resp-1> attempt to resolve the conflict. However, ATM does not confirm that the trajectory modifications still have alternate trajectory options. As a result, <controller(s) performing Resp-1> are unable to issue trajectory modifications to resolve the collision	**A_1: No** *Assumption: Since ATM is performing both Resp-1 and Resp-3, it is easier to coordinate these two responsibilities because they are being performed by the same controller* **A_2: Yes**	**EC-15: Use of** "confirmation of trajectory modifications" input to prevent loss of separation when resolving a conflict
2	Although feedback about the imminent collision is received, <controller(s) performing Resp-1> are pre-occupied with resolving one set of conflicts and therefore do not attend to the feedback about another imminent collision. As a result, <controller(s) performing Resp-1> does not issue trajectory modifications to resolve the imminent collision	**A_1: Yes** **A_2: No** *Assumption: Even if some aircraft are preoccupied with resolving a conflict, the new aircraft can identify the conflict and coordinate its own set of trajectory modifications if traffic conditions are sufficiently light*	**EC-1: Frequency and complexity** of trajectory modifications decisions to prevent loss of separation when resolving a conflict
3	Although feedback about the imminent collision is received, <controller(s) performing Resp-1> are unable to determine possible movement options due to a component failure that prevents trajectory modifications from being computed. As a result, no trajectory modifications are issued	**A_1: Yes** **A_2: No** *Assumption: With multiple aircraft sharing responsibility for preventing conflicts, a component failure on one of the aircraft should not compromise the ability of other aircraft to prevent conflicts*	**EC-17: Vulnerability** of providing trajectory modifications to prevent loss of separation when a component failure compromises decision making

ID	**Scenario**	**Scenario Occurs?**	**Evaluation Criteria**
4	Although feedback about the imminent collision is received, it takes so long for <controller(s) performing Resp-1> to identify a conflict-free solution that no trajectory modification is issued before the collision occurs. This could occur if, the airspace is so densely populated that resolving a limited initial conflict requires changes to many aircraft trajectories	**A_1: No** *Assumption: Although not fully resolved, it is assumed that ATM would be most likely to have the resources to resolve a conflict in densely populated airspace* **A_2: Yes**	**EC-5: Responsiveness** of trajectory modification decisions to prevent loss of separation when resolving a multi-aircraft conflict in densely populated airspace
5	<Controller(s) performing Resp-1> do not receive feedback of the imminent collision because at the time that it checked the trajectories of all aircraft, their trajectories were not in conflict. However, while resolving another conflict, the trajectories of these aircraft do become in conflict and this is not noticed/resolved until after the previous set of conflicts are resolved.	**A_1: Yes** **A_2: No** *Assumption: Even if some aircraft are preoccupied with resolving a conflict, the new aircraft can identify the conflict and coordinate its own set of trajectory modifications if traffic conditions are sufficiently light*	**EC-4: Ability** to make appropriate trajectory modification decisions to prevent loss of separation when multiple conflicts occur
6	<Controller(s) performing Resp-1> does not receive feedback of the imminent collision but the imminent collision is present. This could occur if it does not receive timely feedback on the presence of new ground hazards (e.g., a new construction crane). As a result, it does not believe a collision is imminent and does not try to modify aircraft trajectories to avoid the collision	**A_1: Yes** **A_2: No** *Assumption: Even if some aircraft are preoccupied with resolving a conflict, the new aircraft can identify the conflict and coordinate its own set of trajectory modifications if traffic conditions are sufficiently light*	**EC-12: Timeliness** of ground hazards feedback to prevent loss of separation when resolving a conflict involving terrain or ground obstacles

ID	Scenario	Scenario Occurs?	Evaluation Criteria
7	Trajectory modifications are provided by <at least one of the controller(s) performing Resp-1> and received by the aircraft but the conflict is not resolved. This could occur if the aircraft is preoccupied with other flight deck tasks and does not attend to the trajectory modification issued to it and therefore does not recognize that trajectory modifications have been received	**A_1: Yes** **A_2: No** *Assumption: since the aircraft are coordinating to select trajectory modifications, they know those modifications are coming and will be more responsive in executing them once they are selected.*	**EC-19: Responsiveness of execution** of trajectory modifications to prevent loss of separation when trajectory modifications have been issued
8	Trajectory modifications are provided by <one of the controller(s) performing Resp-1> and received by the aircraft but the conflict is not resolved. This could occur if one of the aircraft receives the trajectory modification but does not execute the trajectory modification because they wrongly believe that the provided trajectory modification would result in another violation of minimum separation. They therefore ignore the provided trajectory modification and make an independent decision which ultimately violates minimum separation	**A_1: Yes** **A_2: No** *Assumption: It is assumed that since the aircraft are selecting their own trajectory modifications, they are therefore already aware of how those trajectory modifications were chosen*	**EC-9: Situational awareness** of trajectory modifications rationale to prevent loss of separation when receiving trajectory modifications to execute
9	Although the imminent collision is recognized, <controller(s) performing Resp-1> gets repeatedly interrupted by changing flight conditions and it constantly needs to modify its solution. As a result, a final solution is not selected until it is too late to prevent a collision	**A_1: No** *Assumption 1: ATM will not have to coordinate conflicts as frequently because it has broader situational awareness of the airspace and can resolve conflicts in a more coordinated fashion.* *Assumption 2: Operators will notify ATM of flights with more advance notice, allowing ATM to pre-coordinate those flights* **A_2: Yes**	**EC-6: Responsiveness** of trajectory modifications decisions to prevent loss of separation when the state of the airspace changes rapidly

ID	Scenario	Scenario Occurs?	Evaluation Criteria
10	Although <controller(s) performing Resp-1> recognize the imminent collision, the process of having possible trajectory modifications confirmed by ATM is cumbersome enough that by the time the trajectory modifications have been confirmed, they are issued too late to adequately prevent the collision	**A_1: No** *Assumption: Since Resp-3 and Resp-1 are both performed by ATM, this coordination between responsibilities can happen much faster than it would between aircraft* **A_2: Yes**	**EC-15: Use** of "confirmation of trajectory modifications" input to prevent loss of separation when resolving a conflict
11	<Controller(s) performing Resp-1> do not receive feedback on time that indicates a collision is imminent because of a degradation in the aircraft's navigational capabilities. If the aircraft does not report this degradation in a timely manner, there may not be enough time to select appropriate trajectory modifications to resolve the conflict	**A_1: Yes** **A_2: No** *Assumption 1: The aircraft will be directly aware of their own flight conditions* *Assumption 2: Coordinating between aircraft on navigational capabilities is faster than coordinating with ATM*	**EC-14: Timeliness** of aircraft capabilities, flight conditions and operational constraints feedback to prevent loss of separation when resolving a conflict
12	Trajectory modifications are received by the aircraft on time but the aircraft does not execute those modifications on time because they were not expecting to receive trajectory modifications (e.g., in a critical phase of flight) and therefore is delayed more than expected in executing the new trajectory.	**A_1: Yes** **A_2: No** *Assumption: Since the aircraft are coordinating trajectory modifications, they would know about potential collisions that they need to resolve and thus this scenario would not occur*	**EC-19: Responsiveness** of execution of trajectory modifications to prevent loss of separation when trajectory modifications have been issued

ID	Scenario	Scenario Occurs?	Evaluation Criteria
13	An emergency or last-minute airspace restriction occurs, requiring large numbers of aircraft to modify their trajectories. As a result, <controller(s) performing Resp-1> are forced to quickly modify trajectories for numerous aircraft before they can adequately consider the collision implications of the new trajectories. As a result, they select the trajectory modifications that result in the fewest collisions and issues those even though some trajectories have collisions.	**A_1: No** *Assumption: ATM will be able to coordinate large groups of aircraft trajectories more easily because of its broader situational awareness of the airspace state* **A_2: Yes**	**EC-10: Level of situational awareness** of airspace state available to prevent loss of separation when trajectory modifications must be identified under challenging or extremely limiting trajectory constraints
14	Although <controller(s) performing Resp-1> receives feedback indicating that the trajectory modifications will result in collision, it believes that it can issue further trajectory modifications later to prevent collision. However, <controller(s) performing Resp-1> do not return to correct that collision (e.g., because it becomes preoccupied resolving other collisions) and thus the collision occurs	**A_1: Yes** **A_2: No** *Assumption: The aircraft will have fewer collisions to attend to and therefore are less likely to be unable to return to a secondary collision to resolve it*	**EC-1: Frequency and complexity** of trajectory modifications decisions to prevent loss of separation when resolving an urgent conflict
15	<Controller(s) performing Resp-1> do not receive timely feedback that the trajectory modifications will result in a collision because they do not receive timely feedback of ground hazards and wrongly believe that the trajectory modifications they are providing are not in conflict with a ground hazard	**A_1: Yes** **A_2: No** *Assumption: The aircraft will have appropriate sensors to be able to detect ground hazards with enough range to take action to avoid them*	**EC-12: Timeliness** of ground hazards feedback to prevent loss of separation when resolving a conflict involving terrain or ground obstacles

ID	Scenario	Scenario Occurs?	Evaluation Criteria
16	<controller(s) performing Resp-1> do not receive timely information about the aircraft's capabilities and, for example, could wrongly believe that an aircraft is capable of more precise navigation than it is. As a result, <controller(s) performing Resp-1> provides trajectory modifications that it wrongly believes does not contain collisions	**A_1: Yes** **A_2: No** *Assumption 1: The aircraft will be directly aware of their own flight conditions and navigational capabilities.* *Assumption 2: Coordinating between aircraft on flight conditions and aircraft capabilities is faster than coordinating with ATM*	**EC-14: Timeliness** of aircraft capabilities, flight conditions and operational constraints feedback to prevent loss of separation when resolving a conflict
17	<Controller(s) performing Resp-1> issue trajectory modifications that do not result in collision. However, during transmission to the aircraft, a communications error occurs, and the aircraft only receives part of the trajectory modifications and the part that is received is in collision with another aircraft trajectory	**A_1: Yes** **A_2: No** *Assumption: It is assumed that a compromised communication link experienced by one aircraft will typically not affect all aircraft (some exceptions exist)*	**EC-18: Vulnerability** of providing trajectory modifications to prevent loss of separation when errors with the communications path occurs
18	<Controller(s) performing Resp-1> receives feedback that the aircraft is not following its trajectory exactly as expected but does not receive enough feedback to know exactly how much extra space buffer to provide. As a result, <controller(s) performing Resp-1> provides trajectory modifications to allow extra spacing between aircraft even though they might be unnecessary	**A_1: Yes** **A_2: No** *Assumption: The aircraft are aware of their own flight conditions and can determine their navigational capabilities with better accuracy to determine how much spacing from other aircraft is needed*	**EC-14: Timeliness** of aircraft capabilities, flight conditions and operational constraints feedback to prevent loss of separation when resolving a conflict

ID	Scenario	Scenario Occurs?	Evaluation Criteria
19	<Controller(s) performing Resp-1> does not receive timely information about the aircraft capabilities. As a result, <controller(s) performing Resp-1> has the wrong belief about the capabilities of the aircraft when they change over time. For example, an aircraft might experience temporary degradation of their GPS signal that recovers after a short time. As a result, <controller(s) performing Resp-1> have wrong/outdated beliefs about the navigational capabilities of the aircraft and issues trajectories with unnecessarily large spacing/clearance	**A_1: Yes** **A_2: No** *Assumption: Since aircraft are coordinating their trajectory modifications, they will be motivated to optimize the trajectories once any degradation is resolved.*	**EC-3: Responsiveness** of trajectory modifications decisions to prevent inability to complete missions when providing more spacing between aircraft due to degraded navigational capabilities
20	<Controller(s) performing Resp-1> do not receive feedback indicating that the trajectories will consume more space than necessary because they originally believed that there was extra airspace available to increase the separation between flights. However, a later flight is filed whose optimal flight path conflicts with the first. Instead of modifying the trajectory to just what is necessary, the later flight's flight plan is modified instead because <controller(s) performing Resp-1> do not recognize that the first flight's trajectory is consuming more space than necessary	**A_1: No** *Assumption: If ATM makes these decisions centrally, it maintains awareness of where it has chosen to consume more airspace than necessary in its trajectory modifications so that those decisions can be undone if necessary* **A_2: Yes**	**EC-7: Responsiveness** of trajectory modifications decisions to enable aircraft to complete missions when reducing spacing between aircraft to accommodate additional air traffic

ID	Scenario	Scenario Occurs?	Evaluation Criteria
21	<Controller(s) performing Resp-1> do not receive feedback indicating that the trajectories will consume more airspace than necessary because the aircraft try to reserve more airspace than necessary for themselves as a safety margin and therefore indicate that they are capable of less precise navigation than they actually are. As a result, <controller(s) performing Resp-1> bases its selection of trajectory modifications on that feedback, not realizing that those trajectories consume more airspace than necessary	**A_1: No** *Assumption: Since ATM is centrally responsible, it can balance the requests and needs of each aircraft with the resources (e.g., airspace, time) available for flights and ensure fair allocation of those resources* **A_2: Yes**	**EC-7: Responsiveness** of trajectory modifications decisions to enable aircraft to complete missions when preventing unnecessary increases in spacing for additional safety margin
22	<Controller(s) performing Resp-1> begin to modify the trajectories of active flights to accommodate a higher-priority flight. However, while selecting these trajectory modifications, the higher-priority flight changes its flight plan. To avoid the additional workload of re-selecting those trajectory modifications, <controller(s) performing Resp-1> may choose to just provide the already-selected trajectory modifications instead of re-selecting them even though they consume more airspace than necessary.	**A_1: Yes** **A_2: No** *Assumption: The aircraft would be motivated to modify their trajectories to be more efficient if such an option were available.*	**EC-2: Responsiveness** of trajectory modifications decisions to prevent inability to complete missions when a high-priority flight changes its planned trajectory

ID	Scenario	Scenario Occurs?	Evaluation Criteria
23	Although <controller(s) performing Resp-1> recognize that operational constraints would be exceeded (e.g., a significant extension of the flight path), it believes that it would be optimal to minimize the operational impacts to other aircraft (that might have fly at higher speeds or be carrying more people) and as a result saddles a single aircraft with numerous operational impacts (e.g., multiple flight path extensions or delays)	**A_1: No** *Assumption: With ATM's centralized role in this architecture, it would be better able to balance equitable distribution of operational impacts amongst aircraft* **A_2: Yes**	**EC-11: Level of situational awareness** of operational impacts occurred by each flight to prevent inability to complete missions when distributing operational impacts over numerous flights
24	<Controller(s) performing Resp-1> does not receive feedback that the trajectory modification would not meet operational constraints because some operational constraints change over time (e.g., diversion options become more restricted due to fuel remaining as a flight progresses). If <controller(s) performing Resp-1> does not receive timely feedback indicating a change to the operational constraints when it selects alternative trajectories, it could select trajectory modifications based on out-of-date or incomplete operational constraints.	**A_1: Yes** **A_2: No** *Assumption: Since the aircraft are responsible for conflict avoidance, they would be aware of changes and would initiate trajectory modifications if needed.*	**EC-13: Timeliness of operational** constraints feedback to prevent loss of separation when operational constraints are changing frequently
25	<Controller(s) performing Resp-1> recognize that UAM aircraft that will interfere with the flight of an emergency response aircraft. However, especially if the airspace is densely occupied, <Controller(s) performing Resp-1> may not be able to clear a path for the emergency response aircraft in time before the emergency response aircraft needs to depart.	**A_1: No** *Assumption: Because ATM has broader SA of the airspace, it can clear a path for an emergency response aircraft faster than the aircraft could individually* **A_2: Yes**	**EC-8: Responsiveness** of trajectory modification decisions to inability to complete missions when a high-priority flight needs to be given precedence for mission completion

ID	Scenario	Scenario Occurs?	Evaluation Criteria
26	<Controller(s) performing Resp-1> receives feedback at the last minute indicating that UAM aircraft are interfering with the flight of an emergency response aircraft because the emergency response aircraft are modifying their trajectories quickly in response to an evolving emergency event (e.g., a bad accident, a wild fire). If <controller(s) performing Resp-1> is unable to respond and modify the trajectories of UAM aircraft in response, they can end up interfering with the emergency response flight	**A_1: No** *Assumption: Because ATM has broader situational awareness of the airspace, it can clear a path for an emergency response aircraft faster than the aircraft could individually* **A_2: Yes**	**EC-8: Responsiveness** of trajectory modification decisions to inability to complete missions when <u>a high-priority flight needs to be given precedence for mission completion</u>
27	<Controller(s) performing Resp-1> does not receive feedback indicating that the trajectory of an aircraft conflicts with an obstacle or terrain. This might occur if the obstacle is new and/or temporary (e.g., a tall crane or a temporary structure near a UAM aerodrome) and feedback about the presence of this obstacle has not been provided to <controller(s) performing Resp-1>.	**A_1: Yes** **A_2: No** *Assumption: The aircraft would have onboard sensing capable of detecting the obstacle with enough range to allow time for the aircraft to respond to avoid a collision with the ground hazard*	**EC-12: Timeliness** of ground hazards feedback to prevent loss of separation when <u>resolving a conflict</u>
28	<Controller(s) performing Resp-1> receive feedback that the aircraft require an immediate change in trajectory but is unable to issue trajectory modifications in time because it must coordinate the change with numerous other aircraft trajectories. This is especially challenging when the airspace is densely populated. However, it is unable to issue the required trajectory modifications in time	**A_1: No** *Assumption: Since ATM has broader situational awareness of the airspace, it can solve large conflict sets faster than the aircraft could individually* **A_2: Yes**	**EC-16: Use of** "mutual agreement" input to prevent loss of separation when <u>resolving a conflict involving numerous aircraft and/or densely populated airspace</u>

ID	Scenario	Scenario Occurs?	Evaluation Criteria
29	<Controller(s) performing Resp-1> correctly recognize that the trajectory needed by an aircraft experiencing an emergency conflicts with those of other aircraft. However, they are unable to select appropriate trajectory modifications because multiple aircraft have various operational constraints that <controller(s) performing Resp-1> is unaware of until it requests for feedback about them. By the time it has received this feedback, there is not enough time to issue appropriate trajectory modifications to resolve the conflict	**A_1: No** *Assumption: As the central decision maker, ATM can more quickly gather and account for the various operational constraints to select trajectory modifications* **A_2: Yes**	**EC-6: Responsiveness** of trajectory modifications decisions to prevent loss of separation when <u>resolving a conflict involving restrictive operational constraints</u>
30	<Controller(s) performing Resp-1> receive feedback indicating that there is a more efficient or desirable trajectory available for an aircraft but does not issue trajectory modifications. This could occur if <controller(s) performing Resp-1> decides to avoid the extra workload of switching the aircraft over to the more efficient trajectory and decides to just leave the aircraft on its less efficient or desirable trajectory	**A_1: Yes** **A_2: No** *Assumption: The aircraft would be motivated to modify their trajectories to be more efficient if such an option were made available*	**EC-2: Responsiveness of** trajectory modifications decisions to prevent inability to complete missions when <u>a high-priority flight is no longer being performed</u>

Appendix E Design Iteration 2 – Analysis of Shared Responsibility Architecture

This appendix presents the STPA-Teaming results from analyzing the shared responsibility collision avoidance architecture chosen in design iteration 1. As described in Section 5.2, STPA-Teaming was used to analyze the *Trajectory Modifications* control action to determine how ATM and the aircraft collectively providing or not providing trajectory modifications could lead to unsafe behavior. Four main combinations of control actions are considered:

- **Combination 1:** Neither ATM nor the aircraft provide trajectory modifications when….
- **Combination 2:** Either ATM or the aircraft provide trajectory modifications when….
- **Combination 3:** Both ATM and the aircraft provide trajectory modifications when…..
- **Combination 4:** One controller (ATM or the aircraft) provides trajectory modifications before the other provides trajectory modifications when….

In the following subsections, the abstract and refined UCCAs and scenarios identified for each of these combinations are presented. The hazards associated with each UCCA are linked in square braces. In addition, the UCCAs for which scenarios were generated are colored in blue. For the causal scenarios, the requirement derived from each scenario is linked in square braces.

E.1. Combination 1 UCCAs and Scenarios

Table E-1: Combination 1 UCCAs and Scenarios

ID	UCCA
UCCA-1	Neither ATM nor the aircraft provide trajectory modifications when the trajectories of two aircraft are in conflict [H-1]
UCCA-2	Neither ATM nor the aircraft provide trajectory modifications when UAM aircraft interfere with the flight of an emergency response aircraft [H-3]
UCCA-3	Neither ATM nor the aircraft provide trajectory modifications when the trajectory of an aircraft conflicts with an obstacle or terrain [H-1]
UCCA-4	Neither ATM nor the aircraft provide trajectory modifications when the trajectory needed by an aircraft experiencing an emergency conflicts with other aircraft [H-1, H-3]
UCCA-5	Neither ATM nor the aircraft provide trajectory modifications when the arrival trajectory of a UAM aircraft at a conventional airport conflicts with the approach course used by conventional aviation aircraft [H-1, H-3]
UCCA-6	Neither ATM nor the aircraft provide trajectory modifications when a higher priority flight incurs an unacceptable operational impact (e.g., delay) due to UAM flights that are occurring [H-3]
UCCA-7	Neither ATM nor the aircraft provide trajectory modifications when UAM aircraft need to be sequenced for arrival to a conventional airport [H-3]
UCCA-8	Neither ATM nor the aircraft provide trajectory modifications when UAM aircraft have overlapping arrival or departure trajectories [H-1, H-3]
UCCA-9	Neither ATM nor the aircraft provide trajectory modifications when the trajectory of a UAM aircraft will take it toward inclement weather that exceeds the aircraft's capabilities [H-1, H-3]

<u>Scenarios for UCCA-1:</u> Neither ATM nor aircraft provide trajectory modifications when the trajectories of two aircraft are in conflict [H-1]

CS-2.1.1. ATM and the aircraft both receive feedback about the imminent collision but none of them issue trajectory modifications to resolve the conflict. This could occur if:

CS-2.1.1-1. The aircraft attempt to resolve the conflict and ATM does not. However, when the aircraft attempt to verify with ATM that their selected trajectory modifications will have alternate trajectory options available, ATM does not confirm this and therefore the aircraft are unable to issue trajectory modifications before the collision occurs. [↓Req-113]

CS-2.1.1-2. ATM allows the aircraft to resolve the conflict. However, if the conflict involves a large number of aircraft or numerous operational constraints, it may take too long for the aircraft to coordinate among themselves to resolve the collision. As a result, neither ATM nor the aircraft issue trajectory modifications to prevent the conflict. [↓ Req-101, Req-102, Req-103]

CS-2.1.1-3. ATM is preoccupied with a previous conflict and the aircraft assume ATM will resolve the conflict and therefore do not resolve the conflict themselves. As a result, none of them issue trajectory modifications to prevent the conflict. [↓ Req-101, Req-102, Req-103]

CS-2.1.1-4. ATM and the aircraft both assume the other is better equipped to resolve the conflict or they each wrongly believe the other will resolve the conflict. As a result, each waits for the other to resolve the conflict and neither of them selects trajectory modifications to prevent the conflict. [↓ Req-121]

CS-2.1.1-5. Although the imminent collision is recognized by both ATM and the aircraft, neither is able to identify trajectory modifications before the collision occurs. For ATM, this could occur if it is resolving a conflict set involving a large number of aircraft. Similarly, for the aircraft, if a large number of aircraft are involved, the coordination required amongst aircraft may slow down conflict resolution. [↓ Req-101, Req-102]

CS-2.1.1-6. Although both ATM and the aircraft receive feedback about the imminent collision, ATM wrongly omits the track or trajectory of one of the aircraft from the consolidated airspace state that it provides to the aircraft to support collision avoidance. This could occur either maliciously (e.g., a bad actor deleting trajectory data) or unintentionally (e.g., data errors). As a result, although the aircraft receive correct feedback about the imminent collision, they choose to ignore that correct feedback because they wrongly believe that ATM's consolidated airspace state information is valid. As a result, both ATM and the aircraft do not attempt to resolve the imminent collision. [↓ Req-111, Req-118,]

CS-2.1.1-7. Although the imminent collision is recognized by both ATM and the aircraft, both are delayed by ATM needing to confirm those trajectory modifications. This could be especially likely to happen if ATM and the aircraft both try to verify trajectory modifications to resolve the same conflict. As a result, ATM does not confirm those trajectory modifications and neither is able to select trajectory modifications in time. [↓ Req-113, Req-121]

CS-2.1.2. ATM and the aircraft do not receive feedback about the imminent collision because:

CS-2.1.2-1. ATM and the aircraft receive inaccurate of out-of-date information about the navigational capabilities of each aircraft. As a result, if one aircraft's navigational capabilities changes (e.g., becomes degraded), neither ATM nor the aircraft receive timely feedback about the change. They are therefore unaware that the aircraft with altered navigational capabilities is on a collision course with another aircraft even though it is. [↓ Req-111]

CS-2.1.3. Either ATM or the aircraft recognize a potential conflict and attempt to provide trajectory modifications to prevent it. However, those trajectory modifications are not received by the aircraft because:

CS-2.1.3-1. Malicious interference or equipment failure results in trajectory modifications not reaching the aircraft. As a result, the aircraft attempt to resolve conflicts independently and select trajectory modifications that conflict with those selected by other aircraft. [↓ Req-104]

CS-2.1.4. ATM and the aircraft both receive feedback about the potential conflict and at least one of them correctly issues trajectory modifications to resolve it. However, the collision still occurs. This could occur if:

CS-2.1.4-1. While the aircraft are still trying to identify appropriate trajectory modifications, ATM selects one and transmits it to the aircraft. However, the aircraft are preoccupied with identifying their own resolution to the conflict and do not recognize that ATM has transmitted trajectory modifications to them already. As a result, they either do not execute them at all or execute them too late for the modifications to be effective. [↓ Req-121, Req-112]

CS-2.1.4-2. The aircraft do execute the trajectory modifications provided by ATM but not to the required level of performance (e.g., precision). This could occur either due to inappropriate equipage, inadequate maneuvering capability or environmental conditions interfering with the flight (e.g., wind gusts, temporary GPS outage). As a result, the provided trajectory modifications do not adequately resolve the conflict. In addition, if ATM notices this too late, there may not be enough time to recompute new trajectory modifications to resolve the conflict. [↓ Req-111, Req-145]

CS-2.1.4-3. The aircraft is deliberately ignoring the trajectory modification provided by ATM and continues on its original trajectory or a different one while other aircraft are executing the trajectory modifications provided to them. As a result, new conflicts arise. [↓ Req-107]

CS-2.1.4-4. The aircraft execute trajectory modifications that are either provided by ATM or selected by them but these trajectory modifications do not adequately resolve the collision. ATM notices this but believes that the aircraft should attempt to re-resolve the collision themselves. However, by the time ATM has notified the aircraft to re-resolve the potential collision, there is not enough to identify new trajectory modifications before the collision occurs. [↓ Req-109]

<u>**Scenarios for UCCA-2:**</u> Neither ATM nor the aircraft provide trajectory modifications when UAM aircraft interfere with the flight of an emergency response aircraft [H-3]

CS-2.2.1. ATM and the aircraft both recognize that a UAM flight will interfere with that of an emergency response aircraft. However, neither of them issue trajectory modifications to address the interference. This could occur because:

CS-2.1.1-1. Especially if the airspace is densely occupied, ATM may be preoccupied resolving more imminent conflicts and the aircraft may require a long time to adequately coordinate on trajectory modifications to avoid interference. As such, neither ATM nor the aircraft can respond and select trajectory modifications before the emergency response aircraft needs to depart. [↓ Req-101, Req-102]

CS-2.1.1-2. ATM and the aircraft receive the feedback about the interference at the last minute because the emergency response flight needs to depart immediately or it is changing its flight path in response to a rapidly evolving emergency. As a result, ATM and the aircraft are unable to respond and modify aircraft trajectories in time to avoid interference with the emergency response flight. [↓ Req-101, Req-102, Req-122]

CS-2.2.2. ATM and the aircraft do not receive feedback that the trajectory of a UAM aircraft will interfere with the flight of an emergency response aircraft. This might occur because:

CS-2.1.2-1. The emergency response aircraft is changing trajectories rapidly in response to an evolving situation and therefore its desired trajectory is not known in advance. As a result, the emergency response aircraft does not have a planned trajectory that can be used to inform conflict resolution decisions. [↓ Req-128]

__Scenarios for UCCA-3:__ Neither ATM nor the aircraft provide trajectory modifications when the trajectory of an aircraft conflicts with an obstacle or terrain [H-1]

CS-2.3.1. Feedback is received that indicates that the trajectory of an aircraft conflicts with an obstacle or terrain but neither ATM nor the aircraft issue trajectory modifications to resolve it. This could occur because:

CS-2.3.1-1. ATM either assumes that the aircraft will resolve the conflict with terrain or it does not receive the feedback about the aircraft's conflict with terrain and therefore takes no action. However, due to other traffic close to the aircraft and the aircraft's proximity to the terrain, the aircraft are unable to select appropriate trajectory modifications before a collision occurs. [↓ Req-108]

CS-2.3.1-2. The aircraft correctly recognize the conflict and request ATM's assistance to resolve the conflict. However, because ATM is dependent on the aircraft to receive feedback about the obstacle or terrain they conflict with, ATM does not receive that feedback with enough time to issue trajectory modifications to prevent a collision before it occurs. [↓ Req-105]

CS-2.3.1-3. The aircraft whose trajectory conflicts with an obstacle or terrain tries to resolve the conflict by communicating with other nearby aircraft to coordinate a trajectory change. However, one of the nearby aircraft is not equipped appropriately to coordinate trajectory modifications. By the time the original aircraft notifies ATM to assist in the conflict, there is not enough time to coordinate amongst the aircraft to resolve the conflict before it occurs. [↓ Req-114]

CS-2.3.4. Either ATM or the aircraft correctly select trajectory modifications to resolve an aircraft's conflict with terrain or an obstacle. However, a collision with an obstacle or terrain still occurs. This could occur if:

CS-2.3.4-1. The first aircraft was successfully able to prevent a collision with an obstacle or terrain. However, a second aircraft selects trajectory modifications without being aware of the terrain or obstacle that the first aircraft had successfully avoided. As a result, the second aircraft selects trajectory modifications that result in a collision with the obstacle or terrain. [↓ Req-116]

<u>**Scenarios for UCCA-4:**</u> Neither ATM nor the aircraft provide trajectory modifications when the trajectory needed by an aircraft experiencing an emergency conflicts with other aircraft [H-1, H-3]

CS-2.3.4. Feedback is received that indicates that the trajectory needed by an aircraft experiencing an emergency conflicts with other aircraft. However, neither ATM nor the aircraft issue trajectory modifications to resolve the conflict. This could occur because:

CS-2.3.4-1. ATM believes that the aircraft can identify trajectory modifications on their own and therefore allow the aircraft to do so. However, either due to traffic density or tight operational constraints (e.g., fuel), the aircraft are unable to identify appropriate trajectory modifications in time. [↓ Req-103]

E.2. Combination 2 UCCAs and Scenarios

Table E-2: Combination 2 Abstracted UCCAs

#	Ci	Cj(s)	Context
UCCA-10	Trajectory modifications	¬Trajectory modifications	when the trajectory of an aircraft is already valid and optimal [H-3]
UCCA-11	Trajectory modifications	¬Trajectory modifications	when the modifications will result in a secondary collision with another aircraft [H-1]
UCCA-12	Trajectory modifications	¬Trajectory modifications	when those trajectories allocate more airspace than necessary to prevent collisions [H-3]
UCCA-13	Trajectory modifications	¬Trajectory modifications	when the operational constraints for at least one of the aircraft will be exceeded [H-1, H-3]
UCCA-14	Trajectory modifications	¬Trajectory modifications	when the modifications will cause a collision with an obstacle or terrain [H-1]
UCCA-15	Trajectory modifications	¬Trajectory modifications	when the modification requires an aircraft to traverse adverse weather that it is not equipped to handle [H-1]
UCCA-16	Trajectory modifications	¬Trajectory modifications	when they do not satisfy the priority needs of the aircraft [H-1, H-3]

For each of the abstracted UCCAs in Table E-2, two refined UCCAs are generated and their IDs are formatted as follows:

- IDs that have the format "UCCA-XX.1" are refined UCCAs where ATM provides trajectory modifications but the aircraft do not
- IDs that have the format "UCCA-XX.2" are refined UCCAs where the aircraft provide trajectory modifications but ATM does not

<u>**Scenarios for UCCA-10.1:**</u> ATM provides trajectory modifications (and the aircraft do not) when the trajectory of a UAM aircraft is already valid and optimal [H-3]

CS-2.10.1.1. Although ATM knows that the trajectory of an aircraft is already valid and optimal, it issues trajectory modifications anyway. This could occur because:

CS-2.10.1.1-1. ATM is forced to make trajectory modifications to accommodate higher-priority traffic (e.g., an emergency response flight). As a result, it makes a trajectory modification that is no longer optimal for the UAM aircraft and that results in a delay or even the UAM aircraft being unable to complete its mission to allow the higher-priority traffic to proceed as requested. [↓ Req-130, Req-131]

CS-2.10.1.1-2. ATM wrongly identifies a potential conflict that will not actually occur. This could happen if ATM receives erroneous or out-of-date trajectory or track data from the aircraft. For example, due to inclement weather, aircraft may be modifying trajectories and ATM may identify a potential conflict based on the unmodified aircraft trajectories. As a result, ATM unnecessarily modifies the trajectories of aircraft away from the valid and optimal one to prevent a conflict that does not actually exist. [↓ Req-119, Req-131]

CS-2.10.1.4. ATM does not provide trajectory modifications but the aircraft deviate from a valid and optimal trajectory. This could occur because:

CS-2.10.1.4-1. The aircraft changes its trajectory intentionally (e.g., modifying its trajectory for an obstacle that was not previously known) or unintentionally (e.g., strong wind pushes the aircraft off course) [↓ Req-111]

<u>**Scenarios for UCCA-11.1:**</u> ATM provides trajectory modifications (and the aircraft do not) when the modifications will result in a collision

CS-2.11.1.1. ATM and the aircraft receive feedback indicating that the trajectory modifications will result in a collision. However, ATM issues its selected trajectory modifications anyway. This could occur if:

CS-2.11.1.1-1. ATM selects trajectory modifications that contain secondary collisions. It does this believing that it would be faster to issue these first and then resolve the secondary collisions later. However, ATM becomes busy resolving these secondary collisions and does not return to at least some of them in time. Furthermore, the aircraft believe ATM will resolve the secondary collisions and therefore don't try to resolve them on its own. [↓ Req-101, Req-102]

CS-2.11.1.1-2. While attempting to identify appropriate trajectory modifications, ATM does not process feedback about changes to the trajectories of other aircraft and therefore does not update its process model to reflect the new trajectories of those aircraft. Similarly, the aircraft may not receive timely feedback about changes in the trajectories of aircraft because they would only receive such feedback indirectly when the consolidated airspace state feedback changes. As a result, ATM selects trajectory modifications that it wrongly believes are collision free but are actually in conflict based on the updated trajectories of other aircraft and the aircraft do not have the timely feedback to recognize that a secondary collision will occur. [↓ Req-108]

CS-2.11.1.2. ATM and the aircraft do not receive feedback that the trajectory modifications will result in a collision. This could occur because:

CS-2.11.1.2-1. ATM does not receive timely feedback of ground hazards (e.g., a new construction crane being erected) and believes that the trajectory modification it is providing will not cause a conflict with that ground hazard. In addition, the aircraft may be unable to detect the ground hazard with enough time to either avoid it themselves or notify ATM to take action. As a result, the aircraft do not select trajectory modifications and ATM issues its selected trajectory modifications, unaware that it will cause a collision. [↓ Req-123]

CS-2.11.1.2-2. Other aircraft are about to but have not yet modified their trajectories and therefore ATM has not received any feedback that the trajectories of some aircraft are about to be modified when it begins to identify its own trajectory modifications. If it does not receive and process feedback later that the trajectories of some aircraft have been modified, ATM will identify trajectory modifications based on the outdated, unmodified aircraft trajectories and therefore identify its own trajectory modifications that it does not realize are in conflict with the modified trajectories of some aircraft. [↓ Req-135, Req-136]

CS-2.11.1.3. ATM issues trajectory modifications that do not result in a collision. However, trajectory modifications that do result in a collision are received by the aircraft. This could occur because:

CS-2.11.1.3-1. During transmission to the aircraft, part of the trajectory modification is dropped (e.g., due to a partial/temporary communications failure). As a result, the aircraft only receives part of the trajectory modifications and the part that is received by the aircraft is in collision with another aircraft trajectory. However, since the aircraft believe that ATM is managing the collision, they do not check the trajectory modification themselves and simply execute it. [↓ Req-134]

Scenario for UCCA-11.2: The aircraft provide trajectory modifications (and ATM does not) when the modifications will result in a collision

CS-2.11.2.1. The aircraft do not receive feedback that the trajectory modifications will result in a collision. This could occur because:

CS-2.11.2.1-1. One set of aircraft are about to but have not yet modified their trajectories and therefore other aircraft have not received any feedback that the trajectories of some aircraft are about to be modified when they begin to identify their own trajectory modifications. If these other aircraft do not receive and process feedback later that the trajectories of some aircraft have been modified, they will identify trajectory modifications based on the outdated, unmodified aircraft trajectories and therefore identify their own trajectory modifications that they do not realize are in conflict with the modified trajectories of some aircraft. [↓ Req-135, Req-136]

<u>Scenarios for UCCA-12.1:</u> ATM provides trajectory modifications (and the aircraft do not) when those trajectories allocate more airspace than necessary to prevent collisions

CS-2.12.1.1. ATM and the aircraft both receive feedback that the trajectory modifications provided to the aircraft will consume more airspace than necessary but ATM issues them anyway. This could occur because:

CS-2.12.1.1-1. Based on feedback about the track of the aircraft, ATM wrongly believes that the aircraft is not adequately following its trajectory and could get too close to another aircraft. ATM therefore decides to issue further trajectory modifications that expand the amount of airspace consumed for that aircraft. However, if the aircraft is actually following its trajectory closely enough that there was no collision risk, it is actually unnecessary to consume the additional airspace. [↓ Req-117, Req-118]

CS-2.12.1.1-2. ATM believes that weather or other event will occur in the near future that will compromise the ability of aircraft to follow more precise trajectory or the ability to track them precisely. As a result, they issue/select these expanded trajectory modifications anyway to protect airspace safety even though they consume more airspace than necessary at the current time. The additional airspace provided for each aircraft provides the aircraft with additional flexibility and safety margin so the aircraft accept the expanded trajectory modifications. However, if the anticipated event does not ultimately occur, these expanded trajectory modifications will not have been necessary at all. [↓ Req-117, Req-118]

CS-2.12.1.2. SC-12.1.2: ATM does not receive feedback that the trajectories will consume more airspace than necessary even though they do. This could occur because:

CS-2.12.1.2-1. ATM does not receive timely information about the aircraft capabilities. As a result, ATM has the wrong belief about the capabilities of the aircraft and continues to issue trajectory modifications under the wrong belief that aircraft capabilities are degraded even though they are no longer degraded. Since ATM chose to resolve the conflict, the aircraft do not coordinate with each other and therefore also do not realize that the trajectories will consume more space than necessary. [↓ Req-111, Req-115]

<u>Scenarios for UCCA-12.2:</u> The aircraft provide trajectory modifications (and ATM does not) when those trajectories allocate more airspace than necessary to prevent collisions

CS-2.12.2.1. The aircraft and ATM both receive feedback indicating that they have selected trajectory modifications that will allocate more airspace than necessary to prevent collisions. However, they select those trajectory modifications anyway. This could occur because:

CS-2.12.2.1-1. The aircraft correctly identify and begin coordinating a resolution. ATM, seeing that the aircraft are coordinating a solution, decides not to step in to resolve the conflict by itself. As a result, ATM does not have the opportunity to recognize when aircraft might be selecting trajectory modifications that consume more airspace than necessary. As a result, if the aircraft try to maintain more separation from other traffic than necessary, they could end up selecting trajectory modifications that consume more airspace than necessary and ATM would not notice that to correct it. [↓ Req-127]

CS-2.12.2.1-2. The aircraft decide to modify their trajectories and know that the trajectory modifications it is issuing will consume more airspace than necessary to prevent collisions but issues them anyway. This could occur if one aircraft experiences flight conditions that it believes will cause the other aircraft to be unable to follow their pre-arranged trajectories accurately enough. If this aircraft assumes this without confirming, it could unilaterally select an expanded trajectory to protect airspace safety, consuming more airspace than necessary. Since the aircraft decided to modify them on their own, ATM does not notice that the trajectories consume more airspace than necessary. [↓ Req-118, Req-116]

E.3. Combination 3 UCCAs and Scenarios

In this research, only one abstract UCCA is considered for combination 3:

UCCA-17: Both ATM and the aircraft provide trajectory modifications when the trajectory modifications conflict [H-1]

As described in Section 5.2, there are three refined UCCAs that can be generated for UCCA-17. These are shown in Table E-3. Following the table are the causal scenarios identified for each of the three refined UCCAs.

Table E-3: Refined UCCAs for UCCA-17

Sub-ID	ATM	Aircraft 1	Aircraft n	Context
UCCA-17.1	<u>Provides</u> Trajectory Modifications	<u>Provides</u> Trajectory Modifications	<u>Does not provide</u> Trajectory Modifications	when the trajectory modifications conflict [H-1]
UCCA-17.2	<u>Does not provide</u> Trajectory Modifications	<u>Provides</u> Trajectory Modifications	<u>Provides</u> Trajectory Modifications	
UCCA-17.3	<u>Provides</u> Trajectory Modifications	<u>Provides</u> Trajectory Modifications	<u>Provides</u> Trajectory Modifications	

<u>Scenarios for UCCA-17.1:</u> ATM and the aircraft both provide trajectory modifications when they conflict with each other [H-1]

CS-2.17.1.1. ATM and the aircraft receive feedback that the trajectory modifications selected by one of them will conflict with those selected by the other. However, they issue their conflicting trajectory modifications anyway. This could occur if:

CS-2.17.1.1-1. Both ATM and a UAM aircraft identify a potential conflict with another aircraft that is not equipped to perform self-separation. The UAM aircraft proceeds to resolve the conflict under the assumption that the other aircraft will not change trajectory and are able to identify a solution first. However, ATM can control that other aircraft (e.g., by coordinating with conventional ATC). Thus, although ATM knows the aircraft has already selected trajectory modifications, ATM provides what it believes to be a better solution but that conflicts with those of the aircraft. [↓ Req-114]

CS-2.17.1.1-2. Both ATM and the aircraft identify a potential conflict and ATM is the first to issue trajectory modifications to the aircraft. Although the aircraft receive those trajectory modifications, they wrongly believe those modifications will not adequately resolve the collision. The aircraft therefore ignore ATM's trajectory modifications and issue their own trajectory modifications that conflict with those selected by ATM. [↓ Req-106]

CS-2.17.1.1-3. Two aircraft correctly recognize a conflict and agree on trajectory modifications to prevent it. Although ATM knows that the other aircraft has already selected a trajectory modification, ATM believes a different solution exists because it is balancing competing factors differently. Thus, ATM issues trajectory modifications anyway to implement its solution even though it conflicts with that selected by the two aircraft. [↓ Req-106]

CS-2.17.1.1-4. The aircraft and ATM both identify a conflict at the same time and both try to resolve it independently. As a result, ATM and the aircraft choose different trajectory modifications to resolve the conflict. If they manage to select trajectory modifications at about the same time, even if one of them receives feedback that the other has already selected trajectory modifications, they may not process that feedback in time before they provide their own (conflicting) trajectory modifications. [↓ Req-110, Req-121]

CS-2.17.1.1-5. Two aircraft correctly recognize a conflict and agree on trajectory modifications to prevent it. At the same time, ATM also begins to try to resolve the conflict. Because of this, although ATM receives the trajectory modifications selected by the aircraft as feedback, it does not process this feedback because it is trying to resolve the conflict. As a result, by the time ATM realizes the aircraft have already selected trajectory modifications, ATM has already transmitted its own trajectory modification that conflicts with the trajectory modification selected by the equipped aircraft. [↓ Req-106, Req-110]

CS-2.17.1.1-6. Multiple sets of conflicts are occurring and ATM and the aircraft have received feedback about them and are attempting to resolve them. While the aircraft are each only attempting to resolve their own local conflict, ATM is resolving all these conflicts together because it believes it can resolve them more efficiently. As a result, although the aircraft have selected trajectory modifications already, ATM provides a conflicting set to the aircraft. [↓ Req-124]

CS-2.17.1.1-7. ATM and the aircraft both receive feedback about a conflict but only the aircraft also realize that they need to modify their trajectory to avoid weather. Thus, while ATM is identifying trajectory modifications to resolve the conflict, the aircraft are resolving the conflict as well as avoiding weather. If they take about the same time to decide on trajectory modifications, the aircraft might end up selecting trajectory modifications that are different from those selected by ATM. [↓ Req-110, Req-121, Req-146, Req-147]

CS-2.17.1.2. ATM does not receive feedback that its trajectory modifications will conflict with those selected by the aircraft. This could occur if:

CS-2.17.1.2-1. Both ATM and the aircraft identify a potential conflict and attempt to resolve it. If they both select trajectory modifications at about the same time, neither ATM nor the aircraft will receive feedback that the other has already selected trajectory modifications before they provide their own. Thus, they both provide trajectory modifications that conflict with each other. [↓ Req-123]

CS-2.17.1.2-2. Two aircraft approach each other but only one of the aircraft is equipped to perform self-deconfliction. That aircraft thus selects a trajectory modification. ATM, however, does not know that the other aircraft has already selected a trajectory modification and it tries to prevent the conflict by transmitting its own trajectory modification that conflicts with the trajectory modification selected by the equipped aircraft. [↓ Req-114, Req-121]

CS-2.17.1.3. ATM and the aircraft do not provide conflicting trajectory modifications but conflicting trajectory modifications are received by the aircraft. This could occur if:

CS-2.17.1.3-1. An aircraft (aircraft A) is involved in a conflict (conflict 1) and begins coordinating to resolve it. While doing so, it becomes involved in another conflict (conflict 2) but cannot attend to conflict 2 until it has resolved conflict 1. However, the aircraft involved in conflict 2 begin coordinating their trajectory modifications. As a result, aircraft A receives two conflicting sets of trajectory modifications, one to resolve conflict 1 and another to resolve conflict 2. [↓ Req-148]

CS-2.17.1.4. The aircraft do not receive conflicting trajectory modifications but the reason for needing trajectory modifications is not resolved or a collision still occurs. *Same scenarios as CS-2.1.4*

<u>**Scenarios for UCCA-17.2:**</u> The aircraft provide trajectory modifications when they conflict with each other [H-1]

CS-2.17.2.1. The aircraft receive feedback that the trajectory modifications selected by each of them are in conflict but they select them anyway. This could occur if:

CS-2.17.2.1-1. Two aircraft correctly recognize a conflict but conflict in their proposals for how to resolve it. This could occur if the aircraft prioritize different types of safety margins and thus select different trajectory modifications to enact those margins. [↓ Req-109, Req-132, Req-133]

CS-2.17.2.1-2. Two aircraft identify the same conflict at the same time and both attempt to resolve the conflict. They both choose the same trajectory modification for themselves and transmit the modification they intend to follow to the other aircraft, assuming that the other aircraft will pick a deconflicted trajectory. If they both make this assumption, they might both pick the same trajectory modification, assuming that the other aircraft will change its trajectory modification. If neither changes its trajectory modification, the aircraft end up executing the conflicting trajectory modifications they originally chose. [↓ Req-149]

CS-2.17.2.1-3. Two aircraft (A & B) identify a conflict. Aircraft A selects a suitable trajectory modification for itself and indicates its proposal to aircraft B. However, due to traffic density, there is not a suitable trajectory available for aircraft B given the trajectory selected by aircraft A and the air traffic in the surrounding airspace. As a result, aircraft B selects a conflicting trajectory modification and requests aircraft A to change its trajectory to deconflict. However, aircraft A does not respond to this request or it has no other options available and thus they select trajectory modifications that conflict. [↓ Req-150]

CS-2.17.2.2. The aircraft do not receive feedback that they are selecting conflicting trajectory modifications. This could occur if:

CS-2.17.2.2-1. Two aircraft each observe a set of conflicts that do not involve each other and they each coordinate with their own group of aircraft to resolve a conflict. Because these two aircraft are resolving different conflicts, they are not coordinating with each other and therefore do not receive feedback that they are selecting trajectory modifications that will conflict with each other (i.e., a secondary conflict) [↓ Req-108]

CS-2.17.2.2-2. An emergency response aircraft believes they do not need to coordinate trajectories with other aircraft because of a TFR that is put in place to protect the airspace for emergency response operations (e.g., a wildfire). It therefore selects trajectories as needed for its operations. However, one of the UAM aircraft does not obey this TFR and selects a trajectory modification that places it within the TFR. If that emergency response aircraft does not communicate its intent to change trajectories, the UAM aircraft may select further trajectory modifications that conflict with the emergency response aircraft. [↓ Req-151]

CS-2.17.2.3. The aircraft do not select conflicting trajectory modifications but conflicting trajectory modifications are received by at least one of the aircraft. This could occur if:

CS-2.17.2.3-1. An aircraft (aircraft A) identifies two conflicts that need to be resolved. Although ATM issues trajectory modifications to resolve conflict 1, Aircraft A coordinates with other aircraft to resolve conflict 2. As a result, aircraft A receives conflicting trajectory modifications even though the ones selected by the aircraft were not conflicting. [↓ Req-148, Req-152]

CS-2.17.2.4. The aircraft do not select conflicting trajectory modifications and those modifications are received correctly by the aircraft. However, the aircraft end up colliding anyway. *Same scenarios as CS-2.1.4*

<u>**Scenarios for UCCA-17.3:**</u> ATM and the aircraft provide trajectory modifications when they all conflict with each other [H-1]

CS-2.17.3.1. ATM and the aircraft receive feedback that they are selecting trajectory modifications that are mutually in conflict but select them anyway. This could occur if:

CS-2.17.3.1-1. The two aircraft correctly recognize a conflict but do not agree on how to prevent it. Furthermore, ATM issues its own trajectory modifications. This could be because it needs to step in to help the two aircraft select an appropriate trajectory modification or because it believes its solution is a better balance of competing factors. Thus, the aircraft and ATM all issue their own trajectory modifications that all conflict with each other. [↓ Req-123]

CS-2.17.3.1-2. Two different conflicts involving different aircraft are identified. The aircraft involved in each conflict resolve their respective conflicts without coordinating with each other. As a result, they select trajectory modifications that conflict with each other. ATM, however, does resolve these conflicts in a coordinated way and issues conflicting trajectory modifications to the aircraft because it believes it has a better coordinated or more efficient solution to the two conflicts. [↓ Req-121, Req-148, Req-152]

CS-2.17.3.2. ATM and the aircraft do not receive feedback that they are selecting conflicting trajectory modifications. This could occur if:

CS-2.17.3.2-1. Two different conflicts involving different aircraft are identified. The aircraft involved in each conflict resolve their respective conflicts without coordinating with each other. As a result, they select trajectory modifications that conflict with each other without realizing that they do. Although the aircraft provide feedback to ATM that they have selected trajectory modifications, that feedback is not received by ATM. However, because ATM provides no feedback when it receives trajectory modifications selected by the aircraft, ATM and the aircraft are all unaware that mutually conflicting trajectory modifications have been selected. [↓ Req-121, Req-148, Req-152]

CS-2.17.3.3. ATM and the aircraft do not select mutually conflicting trajectory modifications but the trajectory modifications received by the aircraft are mutually conflicting. *Same scenarios as CS-2.17.1.3 and CS-2.17.2.3*

CS-2.17.3.4. ATM and the aircraft do not select mutually conflicting trajectory modifications and they are correctly received by the aircraft. However, the aircraft end up colliding anyway. *Same scenarios as SC-2.1.4*

E.4. Combination 4 UCCAs and Scenarios

Two abstract UCCAs were identified for combination 4 and these are shown in Table E-4. As in the previous section, three refined UCCAs were identified for each of these abstract UCCAs and they are shown in Table E-5. Following these tables are the scenarios that were identified for each refined UCCA.

Table E-4: Combination 4 Abstract UCCAs

#	Either ATM or the aircraft	Then the other	Context
UCCA-18	Provides Trajectory Modifications	Provides Trajectory Modifications	when ATM and the aircraft are attempting to resolve the same conflict [H-1]
UCCA-19			When ATM and the aircraft are modifying trajectories for different reasons [H-1]

Table E-5: Refined UCCAs for UCCA-18 and UCCA-19

Sub-ID	Trajectory Modifications provided by	Then trajectory Modifications provided by	Context
UCCA-18.1	ATM	Aircraft n	when ATM and the aircraft are attempting to resolve the same conflict [H-1]
UCCA-18.2	Aircraft n	ATM	
UCCA-18.3	Aircraft 1	Aircraft n	
UCCA-19.1	ATM	Aircraft n	When ATM and the aircraft are modifying trajectories for different reasons [H-1]
UCCA-19.2	Aircraft n	ATM	
UCCS-19.3	Aircraft 1	Aircraft n	

<u>**Scenarios for UCCA-18.1:**</u> ATM provides trajectory modifications then the aircraft provide trajectory modifications when ATM and the aircraft are attempting to resolve the same conflict *[H-1]*

CS-2.18.1.1. One or more of the aircraft provide trajectory modifications after ATM does even though they both received feedback indicating the conflict at the same time. This could occur if:

CS-2.18.1.1-1. When feedback about the conflict is received, both ATM and the aircraft begin trying to resolve it. If the conflict occurs in densely populated airspace, ATM is faster at identifying a solution and transmits its solution to the aircraft. However, the aircraft are preoccupied with identifying a solution and don't process the trajectory modifications from ATM and issue their own trajectory modifications to resolve the same conflict. [↓ Req-121, Req-153]

CS-2.18.1.1-2. When feedback about the conflict is received, both ATM and the aircraft begin trying to resolve it. ATM is faster at identifying a solution and the aircraft correctly receive that solution. However, the aircraft disagree with ATM's selected trajectory modifications and believe they have the authority (e.g., PIC authority) to select different ones and therefore the aircraft select different trajectory modifications for the same conflict. [↓ Req-133, Req-154, Req-155]

CS-2.18.1.1-3. The aircraft have more direct feedback about prevailing flight conditions and recognize that inclement weather or other factors may make it more difficult for aircraft to fly standard trajectories. Thus, the aircraft take additional time to coordinate trajectory modifications to account for these factors. ATM, however, does not realize these conditions are happening and therefore is faster at issuing trajectory modifications because it wrongly believes that standard trajectories can be used. [↓ Req-111, Req-116, Req-121]

CS-2.18.1.2. ATM and the aircraft do not receive feedback indicating the conflict at the same time. This could occur if:

CS-2.18.1.2-1. The aircraft are temporarily unable to receive consolidated airspace state feedback (e.g., terrain or building interference) and therefore get that feedback later than ATM. As a result, ATM resolves the conflict first and then only do the aircraft identify their solution. [↓ Req-156]

CS-2.18.1.3. ATM does not provide trajectory modifications before the aircraft do but the aircraft do receive trajectory modifications from ATM before they select their own. This could occur if:

CS-2.18.1.3-1. ATM modifies the trajectory of the aircraft for some other reason (e.g., a different conflict, weather, traffic) before it realizes a new conflict exists. However, ATM does not indicate that its trajectory modifications are not to resolve that conflict. As a result, for that conflict, although ATM isn't resolving that conflict yet, the aircraft receive trajectory modifications from ATM before they select their own. [↓ Req-132]

CS-2.18.1.4. ATM does not provide trajectory modifications before the aircraft and that is what the aircraft receive. However, they respond by executing a set of trajectory modifications provided by ATM before the ones selected by the aircraft. This could occur if:

CS-2.18.1.4-1. An aircraft receives a set of trajectory modifications from ATM for some other reason and then becomes involved in a conflict and selects trajectory modifications to resolve the conflict (and ATM does not transmit its own trajectory modifications for that conflict). However, if the aircraft apply those modifications in the order they were received and begin to execute the modifications from ATM first before executing the ones selected by the aircraft to resolve the conflict. [↓ Req-157]

__Scenarios for UCCA-18.2:__ The aircraft provide trajectory modifications then ATM provides trajectory modifications when ATM and the aircraft are attempting to resolve the same conflict *[H-1]*

CS-2.18.2.1. ATM provides trajectory modifications after one or more of the aircraft does even though they both received feedback indicating the conflict at the same time. This could occur if:

CS-2.18.2.1-1. In a situation where a large group of aircraft need to alter their trajectories, a subset of the aircraft may select their own trajectory modifications and are able to do so faster than ATM. ATM, however, is identifying a more complete solution for all aircraft which takes it longer but is more optimal/valid. Thus, the aircraft end up selecting one set of trajectories first before ATM identifies its solution. [↓ Req-124]

CS-2.18.2.2. ATM and the aircraft do not receive feedback indicating the conflict at the same time. This could occur if:

CS-2.18.2.2-1. The aircraft receive feedback about each other's trajectory or track via direct transponder links whereas ATM receives that information through a ground receiver network. As a result, ATM receives feedback about the trajectory/track after the aircraft does and therefore the aircraft select trajectory modifications before ATM does. In addition, if ATM does not process feedback that the aircraft have selected trajectory modifications, it won't know to avoid providing its own set of modifications. [↓ Req-121, Req-158]

CS-2.18.2.3. The aircraft do not provide trajectory modifications before ATM does but they receive trajectory modifications selected by them before ATM. This could occur if:

CS-2.18.2.3-1. An aircraft modifies its trajectory for some other reason (e.g., a different conflict, weather, traffic) before it realizes the new conflict exists. Meanwhile, ATM resolves the conflict but the aircraft don't know this until ATM issues its trajectory modifications. As a result, the aircraft receive and execute their own trajectory modification first and only execute ATM's trajectory modifications after. [↓ Req-132, Req-157]

CS-2.18.2.3-2. ATM transmits its trajectory modifications via a ground network to the aircraft whereas the aircraft transmit trajectory modifications directly to each other. As a result, even though they all provide trajectory modifications at the same time, the ones provided by ATM arrive after the ones provided by the aircraft [↓ Req-132, Req-157]

CS-2.18.2.4. The aircraft do not provide trajectory modifications before ATM and that is what they receive. However, they respond by executing trajectory modifications provided by them first before ATM. *Same scenarios as CS-2.18.1.4*

__Scenarios for UCCA-18.3:__ One aircraft provides trajectory modifications then another aircraft provides trajectory modifications when both aircraft are attempting to resolve the same conflict *[H-1]*

CS-2.18.3.1. One aircraft provides trajectory modifications after another aircraft does even though they receive feedback indicating the conflict at the same time. This could occur if:

CS-2.18.3.1-1. Two aircraft are under a heavy workload. Although one aircraft correctly recognizes the conflict and tries to resolve it, the other aircraft does not recognize the conflict immediately and does not try to resolve it until later and does not realize the other aircraft has already tried to coordinate a resolution to the conflict. As a result, the two aircraft initially issue conflicting conflict resolutions to each other. [↓ Req-106, Req-158]

CS-2.18.3.2. The aircraft do not receive feedback indicating the conflict at the same time. This could occur if:

CS-2.18.3.2-1. The aircraft receive feedback about the conflict from different sources. One aircraft might receive the feedback directly via transponder messages from the other aircraft. However, the other aircraft might receive its feedback via a ground receiver network which will have more delay than direct transponder messages. [↓ Req-121]

CS-2.18.3.3. The aircraft do not provide trajectory modifications at different times but they receive trajectory modifications at different times. *Same scenarios as CS-2.18.1.3 and CS-2.18.2.3*

__Scenarios for UCCA-19.1:__ ATM provides trajectory modifications before the aircraft provide trajectory modifications when they are modifying trajectories for different reasons *[H-1]*

CS-2.19.1.1. One or more of the aircraft provide trajectory modifications after ATM even though they receive feedback indicating the need to modify trajectories at the same time. This could occur if:

CS-2.19.1.1-1. ATM and the aircraft receive feedback about weather and a potential conflict at the same time. However, ATM has the wrong process model of the aircraft's ability to operate in inclement weather and believes only a simple trajectory modification is needed to resolve both the conflict and sufficiently avoid the weather. However, the aircraft (which has the correct process model of its own needs) is identifying a more substantial trajectory modification that resolves the conflict and avoids more of the weather. Thus, although ATM is faster at identifying its simpler trajectory modification, the aircraft still select their own trajectory modifications. [↓ Req-111, Req-121, Req-159]

CS-2.19.1.2. ATM and the aircraft do not receive feedback indicating the need to modify trajectories at the same time. This could occur if:

CS-2.19.1.2-1. Both ATM and the aircraft receive feedback about a conflict at the same time but feedback about inclement weather is either only received by the aircraft or received late by ATM. ATM therefore selects trajectory modifications to prevent the conflict only. However, the aircraft select different trajectory modifications that also avoid weather and prevent the conflict. [↓ Req-111, Req-159]

CS-2.19.1.3. ATM and the aircraft do not provide trajectory modifications at different times but the aircraft receive trajectory modifications from the aircraft after ones from ATM. *Same as CS-2.18.1.3*

CS-2.19.1.4. ATM and the aircraft do not provide trajectory modifications at different times. However, the aircraft respond by executing trajectory modifications provided by ATM first before the ones provided by the aircraft. *Same as CS-2.18.1.4*

<u>Scenarios for UCCA-19.2</u> The aircraft provide trajectory modifications then ATM provides trajectory modifications when they are modifying trajectories for different reasons *[H-1]*

CS-2.19.2.1. ATM provides trajectory modifications after one or more of the aircraft do even though they receive feedback indicating the need to modify trajectories at the same time. This could occur if:

CS-2.19.2.1-1. The feedback indicates two conflicts, one more immediate than the other. The aircraft decide to resolve the more immediate one first whereas ATM tries to resolve both conflicts at the same time. Thus, the aircraft identify trajectory modifications more quickly than ATM but ATM issues its trajectory modifications anyway because it has a solution to both conflicts. [↓ Req-121]

CS-2.19.2.2. ATM and the aircraft do not receive feedback indicating the need to modify trajectories at the same time. *Same as CS-2.19.1.2*

CS-2.19.2.3. ATM and the aircraft do not provide trajectory modifications at different times but the aircraft receive trajectory modifications from the aircraft before ones from ATM. *Same as CS-2.18.2.3*

CS-2.19.2.4. ATM and the aircraft do not provide trajectory modifications at different times and that is what they receive. However, the aircraft respond by executing trajectory modifications provided by ATM first before the ones provided by the aircraft. *Same as CS-2.18.2.4*

<u>Scenarios for UCCA-19.3:</u> One aircraft provides trajectory modifications then the other when they are modifying trajectories for different reasons *[H-1]*

CS-2.19.3.1. **One aircraft provides trajectory modifications after the other does even though they receive feedback indicating the need to modify trajectories at the same time.** *Same as CS-2.18.3.1*

CS-2.19.3.2. **ATM and the aircraft do not receive feedback indicating the need to modify trajectories at the same time.** *Same as CS-2.18.3.2*

CS-2.19.3.3. **ATM and the aircraft do not provide trajectory modifications at different times but the aircraft receive trajectory modifications from the aircraft before ones from ATM.** *Same as CS-2.18.3.3*

CS-2.19.3.4. ATM and the aircraft do not provide trajectory modifications at different times. However, the aircraft respond by executing trajectory modifications provided by ATM first before the ones provided by the aircraft. *Same as CS-2.18.3.4*

Appendix F Design Iteration 2 – Requirements and Refined Control Elements

In this appendix, the additional system requirements and the refined control elements that were used to create the iteration 2 shared collision avoidance conceptual architecture (shown in Figure 39) are presented.

F.1 Additional System Requirements for Shared Collision Avoidance

Req-101. If there is not enough time to generate complete trajectory modifications for all aircraft, partial trajectory modifications must be generated that resolve the most imminent conflicts or interference first. [↓ RC-76]

Req-102. If partial trajectory modifications are provided, they must be updated within <TBD> time of issuing them to ensure aircraft have complete and collision-free trajectories to follow for their flight. [↓ RC-77]

Req-103. If either ATM or the aircraft is unable to resolve a potential conflict, the other must be able to take over and resolve it. [↓ RC-78]

Req-104. Conflicts must continue to be resolved even if the ability of ATM or one of the aircraft to do so is compromised [↓ RC-79]

Req-105. Ground hazards must be detected with at least <TBD> range to ensure aircraft can take action to avoid them. [↓ RC-80]

Req-106. The conflict being resolved must be indicated to the aircraft involved to ensure they recognize the collision. [↓ RC-81]

Req-107. If an aircraft does not adequately execute its trajectory modifications, its trajectory should be analyzed with respect to nearby aircraft to identify any potential collisions that might result from the inadequately executed trajectory modification. [↓ RC-82]

Req-108. Any aircraft within <TBD> of an area where a potential conflict might occur or within <TBD> distance of an aircraft whose trajectory is being modified must be included in coordination to ensure secondary collisions are avoided. [↓ RC-83]

Req-109. A potential conflict that remains unresolved after <TBD> of being identified or <TBD> seconds of the potential collision occurring must be prioritized and resolved within<TBD> time. [↓ RC-84]

Req-110. If either ATM or the aircraft decide to attempt to resolve a collision, they must provide feedback of their decision to do so. [↓ RC-85]

Req-111. Any changes to the navigational capabilities of an aircraft (e.g., accuracy) must be reported in a timely manner to ensure that those capabilities are kept up-to-date for use in collision avoidance decisions. [↓ RC-86]

Req-112. The aircraft must begin executing trajectory modifications issued by ATM within <TBD> time of receiving them [↓ RC-87]

Req-113. Requests to confirm that proposed trajectory modifications will have adequate alternative trajectories available must be confirmed within <TBD> time of the request being received to ensure trajectory modifications can be issued in a timely manner. [↓ RC-88]

Req-114. If a conflict involves at least one aircraft that is not equipped or non-cooperative, the conflict must be resolved by whoever has better information about that aircraft. [↓ RC-97]

Req-115. Trajectory modification decisions must account for any flight conditions that cause an aircraft to be unable to meet their expected navigational or maneuvering capabilities or trajectory constraints. [↓ RC-89]

Req-116. If prevailing flight conditions are affecting the ability of one aircraft to meet their expected navigational or maneuvering capabilities or trajectory constraints, the ATM system shall anticipate and verify that other aircraft of similar type might be experiencing the same effects and account for them in trajectory modification decisions. [↓ RC-90]

Req-117. Any trajectory modifications that consume additional airspace must be checked periodically to confirm they are still necessary [↓ RC-91]

Req-118. Trajectories that consume more airspace than necessary must be able to be amended should it become necessary to use airspace more efficiently [↓ RC-92]

Req-119. If a more efficient trajectory becomes available for an aircraft, trajectory modifications must be provided to place the aircraft on the more efficient path within <TBD> time. [↓ RC-95]

Req-120. Any aircraft requiring a change in trajectory for any reason must be able to initiate a request for trajectory modifications. [↓ RC-94]

Req-121. An explicit decision must be made about who is resolving a potential conflict. [↓ Resp-1.2]

Req-122. Emergency response flights must be given priority to carry out their missions. [↓ RC-96]

Req-123. If multiple potential resolutions to a conflict are identified, an explicit decision must be made about which trajectory modification instructions to execute [↓ Resp-1.3]

Req-124. Under <TBD> conditions, to better coordinate the resolution of conflicts, it must be possible to temporarily require that all trajectory modification decisions be made centrally. [↓ Resp-1.6]

Req-125. If the NAS temporarily enters a "centralized mode", it must have an associated end time when that mode ends [↓ RC-100]

Req-126. ATM system shall establish maximum allowable spacing between aircraft based on current and anticipated conditions [↓ RC-101]

Req-127. Flights that require additional spacing beyond <TBD> maximum allowable spacing must be monitored and managed to ensure the additional spacing does not cause undue negative impacts to other airspace users. [↓ RC-102]

Req-128. Emergency response aircraft must be provided with sufficient protected airspace to allow them the flexibility to make some small changes to their trajectory without having to re-coordinate with other aircraft or ATM [↓ RC-104]

Req-129. If a nearby aircraft is detected but not included in the consolidated airspace state provided by ATM, the consolidated airspace state must be reviewed to confirm if the detected aircraft was wrongly omitted. [↓ RC-105]

Req-130. If a more efficient trajectory becomes available for an aircraft, trajectory modifications must be provided to place the aircraft on the more efficient path within <TBD> time [↓ RC-106]

Req-131. Any changes made by the aircraft to its trajectory (e.g., to account for weather) must be accounted for in future trajectory modification decisions [↓ RC-107]

Req-132. Trajectory modifications must always be accompanied by rationale for their selection. [↓ RC-108]

Req-133. When arbitrating conflicting trajectory modifications, the rationale for each trajectory modification selection must be considered [↓ RC-98]

Req-134. Acknowledgements of trajectory modifications must be checked to ensure that they match the originally transmitted trajectory modification [↓ RC-109]

Req-135. When aircraft are identified as needing a trajectory modification, an indicator must be provided within the consolidated airspace state to ensure other aircraft and ATM are aware of aircraft whose trajectories may be changing [↓ RC-110]

Req-136. Trajectory modification decisions must account for aircraft who may be about to change trajectories [↓ RC-93]

Req-137. If a collision is not adequately resolved, the conflict must be re-evaluated and new trajectory modifications issued to resolve it again [↓ RC-103]

Req-138. In consolidated airspace state, indicate if an aircraft is an emergency response aircraft so that additional spacing can be provided for them [↓ RC-111]

Req-139. If multiple conflicting trajectory modifications are issued, none will be transmitted for execution until they are arbitrated [↓ RC-112]

Req-140. Consolidated airspace state must include characteristics of the aircraft and mission along with the trajectory (i.e., aircraft capabilities, mission and operational constraints) [↓ RC-113]

Req-141. Traffic priorities must be accounted for when deciding who should resolve a conflict [↓ RC-114]

Req-142. Selection of trajectory modifications must account for all aircraft that are identified as being potential participants in a conflict [↓ RC-115]

Req-143. The controller selected to resolve the conflict must either attempt to resolve the conflict or indicate that they are unable to [↓ RC-116]

Req-144. Air traffic priorities must be determined and adhered to consistently when making trajectory modification decisions [↓ Resp-1.5]

Req-145. The ability of aircraft to execute their planned trajectory to the required navigational performance must be monitored and trajectory modifications reconsidered if they are unable to execute their planned trajectories sufficiently accurately [↓ Resp-1.4]

Req-146. Aircraft must be able to decline a trajectory modification if the new trajectory cannot be executed safely. [↓ RC-120]

Req-147. If an aircraft declines a trajectory modification, it must provide a reason for declining it [↓ RC-121]

Req-148. If an aircraft is involved in two conflicts at once, a decision must be made about whether these two conflicts should be resolved as two conflicts or if they should be combined into 1 large conflict [↓ RC-122]

Req-149. If the aircraft are resolving a conflict, they must ensure that they select trajectories that do not conflict with each other [↓ RC-123]

Req-150. If aircraft are resolving a conflict, they must be able to provide their trajectory restrictions to the other aircraft to support overall selection of trajectory modifications [↓ RC-124]

Req-151. Even airspace operations that are protected within a TFR must have track and planned trajectories available and kept updated [↓ RC-125]

Req-152. If two conflicts are combined into one, a new decision must be made about which controller resolves that new combined conflict [↓ RC-126]

Req-153. When a controller is selected to resolve the conflict, a time limit for resolving the conflict must be established that is based on how much time is available before a collision occurs [↓ RC-127]

Req-154. Unless an aircraft is unable to execute trajectory modifications, it must execute the trajectory modifications provided to it [↓ RC-128]

Req-155. If an aircraft is unable to execute a provided set of trajectory modifications, it must provide a reason for being unable to execute them so that new trajectory modifications can be selected [↓ RC-129]

Req-156. Feedback mechanisms used by the aircraft to receive feedback needed for collision avoidance must be designed for the anticipated urban environments or terrain conditions of UAM [↓ RC-130]

Req-157. If two different sets of trajectory modifications that modify the same portion of an aircraft's trajectory for two different purposes, they should be arbitrated to decide how to apply both sets of modifications (if possible) [↓ RC-131]

Req-158. If an aircraft receives a message from another aircraft to coordinate trajectory modifications, it must respond with its proposed trajectory modifications within <TBD> time [↓ RC-132]

Req-159. If the aircraft identify a conflict, they must be able to highlight important trajectory constraints to assist with deciding which controller would be best able to resolve the conflict [↓ RC-133]

Req-160. Once a conflict is identified, it must be monitored to ensure it is resolved until there is no longer a risk of a collision. [↓ RC-134]

Req-161. Once a conflict is identified, it must be reported within <TBD> time [↓ RC-135]

Req-162. The controller assigned to resolve a conflict must acknowledge the conflict they are assigned to [↓ RC-136]

Req-163. When switching to fully centralized decision making, a transition period shall be allowed where aircraft continue resolving some conflicts to avoid overwhelming ATM [↓ RC-137]

Req-164. If additional aircraft become involved in a conflict that is already being resolved, the controller chosen to resolve that conflict must be re-evaluated to ensure they are still appropriate. [↓ RC-138]

Req-165. The airspace near an aerodrome must be managed by ATM to protect aircraft entering/exiting the aerodrome unless traffic density levels permit the aircraft to self-separate [↓ RC-139]

Req-166. If a conflict is reassigned to a different controller, controllers that are no longer assigned must stop resolving a conflict [↓ RC-140]

Req-167. If the aircraft are resolving a conflict, they must select trajectory modifications in accordance with the prescribed traffic priorities. [↓ RC-141]

Req-168. If a conflict remains unresolved, the originally assigned controllers must be notified to re-resolve it. [↓ RC-142]

Req-169. If controllers are unable to adequately resolve a conflict after <TBD> attempts, a re-assignment should be considered. [↓ RC-143]

Req-170. Trajectory modifications to resolve a conflict should only be accepted from the controller assigned to the conflict. [↓ RC-144]

Req-171. A controller assigned to a conflict must be given enough time to attempt to resolve the conflict before that conflict is marked as unresolved [↓ RC-145]

Req-172. If an aircraft needs to modify its departure time (earlier or later), it must provide feedback of that for approval and deconfliction with other aircraft [↓ RC-146]

Req-173. If multiple aircraft select the same trajectory modifications, all aircraft other than the one with the highest assigned priority must alter their trajectory modifications [↓ RC-147]

Req-174. The aircraft resolving a conflict must confirm each other's trajectory modifications before they are executed. [↓ RC-148]

Req-175. If two aircraft select similar trajectory modifications and both aircraft are assigned the same priority, they must ensure that one aircraft changes its trajectory modifications [↓ RC-149]

Req-176. A request to take over resolving a conflict must be confirmed with the originally assigned controller before transferring assignment [↓ RC-150]

Req-177. Future trajectory modification decisions must account for any arbitrated trajectory modifications [↓ RC-151]

Req-178. The reference frame used by aircraft to exchange trajectory constraints must be consistent across aircraft [↓ RC-152]

F.2 Refined Control Elements for Shared Collision Avoidance

The figures in this section show how these additional requirements were used in conjunction with the earlier set of system requirements shown in Appendix B to refine Resp-1 and generate the six more detailed control responsibilities and their associated control actions and feedback. Each of these responsibilities and a simplified version of their corresponding control actions and feedback were shown on the refined conceptual architecture shown in Figure 39 in Section 5.3.2. Each control element is traced to the constraint or requirement used to generate it using the links in square braces.

Resp-1.1: Identify and resolve any conflict with an aircraft's trajectory *[Req-4]*

RC-2: Account for planned trajectory when identifying conflicts *[Req-10]*

RC-3: Ensure that coordination provided to the aircraft is within the capabilities of the aircraft *[Req-15]*

RC-4: Ensure coordination decisions do not cause secondary conflicts *[Req-17]*

RC-15: Continue resolving conflicts even if one or more aircraft are unable to communicate or are not responding *[Req-12]*

RC-26: Ensure that aircraft have received the coordination being communicated *[Req-13]*

RC-31: Ensure that alternative movement options are considered when coordinating aircraft *[Req-57]*

RC-54: Ensure that initiated traffic management plans are used to influence trajectory modifications, alternative trajectory selection, and airspace access management *[Req-61]*

RC-58: Confirm alternative trajectories are available for any proposed coordination *[Req-83]*

RC-61: Account for reasons that a trajectory modification was ineffective when selecting new trajectory modifications *[Req-85]*

RC-71: Check in with affected aircraft on preferred trajectory modification if unable to meet all operational constraints *[Req-86]*

RC-80: Detect ground hazards with at least <TBD> range to ensure aircraft can take action to avoid them *[Req-105]*

RC-94: Any aircraft requiring a change in trajectory for any reason must be able to initiate a request for trajectory modifications *[Req-120]*

RC-103: Re-evaluate any inadequately resolved conflict and generate new trajectory modifications *[Req-137]*

RC-112: If conflicting trajectory modifications are issued, only execute a chosen set after arbitration *[Req-139]*

RC-115: Account for all aircraft identified as potential participants in a conflict when selecting trajectory modifications *[Req-142]*

RC-116: Resolve the conflict or indicate if unable if a controller is selected to resolve a conflict *[Req-143]*

RC-151: Account for arbitrated trajectory modifications in future trajectory modification decisions *[Req-177]*

Process Model Parts & Required Feedback/Inputs

Feedback from the aircraft:

- Acknowledgement of traj. mods. *[RC-26]*
- Detected ground hazards *[RC-80]*
- Reason for trajectory deviation *[RC-61]*
- Request for trajectory change *[RC-94]*

Inputs from Resp-1.2:

- Controller assigned to resolve conflict *[RC-116]*
- Aircraft involved in conflict *[RC-15, RC-115]*

Inputs from Resp-1.3: Selected trajectory modifications *[RC-112, RC-151]*

Inputs from Resp-1.4:

- Unresolved Collision Risk *[RC-27, RC-103]*
- Reason for trajectory deviation *[RC-61]*

Inputs from Resp-1, 2, 3, 4 or 5:

- Active traffic mgmt program *[RC-54]*
- Confirm trajectory modifications *[RC-58]*
- Alternate trajectories *[RC-31, RC-58]*
- Aircraft not communicating *[RC-15]*
- Consolidated airspace state *[RC-2, RC-3, RC-4]*

Required Control Actions/Outputs

Control actions to the aircraft:

- Trajectory modifications *[Resp-1]*
- Request acknowl. of traj. mods. *[RC-26]*
- Trajectory modification options *[RC-71]*

Outputs to Resp-1, 2, 3, 4 or 5:

- Trajectory modifications *[Resp-1]*
- Proposed trajectory modifications *[RC-58]*

Figure F-1: Defined control elements for Resp-1.1

Resp-1.2: Decide which controller is resolving a conflict *[Req-121]*

RC-78: Allow ATM and the aircraft to take over from each other to resolve a conflict *[Req-103]*

RC-79: Continue preventing conflicts even if the ability of ATM or one of the aircraft to do so is compromised *[Req-104]*

RC-83: Aircraft within <TBD> of an area where a potential conflict might occur should be included in coordination *[Req-108]*

RC-85: When either ATM or the aircraft decide to resolve a collision, they must notify the other aircraft or ATM of their decision *[Req-110]*

RC-97: If a conflict involves at least one aircraft that is not equipped or non-cooperative, the conflict must be resolved by whoever has better information about that aircraft *[Req-114]*

RC-114: Account for traffic priorities when deciding who should resolve a conflict *[Req-141]*

RC-133: Aircraft must be able to highlight important trajectory constraints to assist with deciding which controller would be best able to resolve the conflict *[Req-159]*

RC-136: Acknowledge the conflict to be resolved once a controller is assigned to resolve the conflict *[Req-162]*

RC-137: Allow for a transition period where aircraft continue resolving some conflicts to avoid overwhelming ATM when initially transitioning to fully centralized decision making *[Req-163]*

RC-143: Consider a re-assignment of controller if a conflict persistently remains unresolved *[Req-169]*

RC-150: A request to take over resolving a conflict must be confirmed with the originally assigned controller *[Req-176]*

Process Model Parts & Required Feedback/Inputs

Feedback from Resp-1.1:

- Identified conflicts *[Resp-1.2]*
- Requested controller to resolve conflict *[RC-78, RC-85]*
- Unable to resolve conflict *[RC-78, RC-79]*
- Ops constraints for identified conflict *[RC-133]*
- Acknowledge conflict to resolve *[RC-136]*
- Assignment transfer accepted *[RC-150]*

Feedback from Resp-1.4:

- Aircraft unable to communicate *[RC-79]*
- Persistent unresolved conflict *[RC-143]*

Inputs from Resp-1.5: Traffic priorities *[RC-114]*

Inputs from Resp-1.6: Implement fully centralized collision avoidance *[RC-137]*

Input from Resp-4: Consolidated airspace state *[RC-97]*

Internal Process Model Variables

- Number of aircraft involved in conflict *[RC-83]*
- Current workload (of controllers resolving conflicts) *[Resp-1.2]*

Required Control Actions/Outputs

Control actions to Resp-1.1:

- Controller assigned to resolve conflict *[Resp-1.2]*
- Aircraft involved in conflict *[RC-83]*
- Proposed assignment transfer *[RC-150]*

Figure F-2: Defined control elements for Resp-1.2

Resp-1.3: Arbitrate any conflicting conflict resolution proposals *[Req-123]*

RC-112: If conflicting trajectory modifications are issued, only execute a chosen set after they have been arbitrated *[Req-139]*

RC-131: Arbitrate two different sets of trajectory modifications that modify the same portion of an aircraft's trajectory for two different purposes to decide how to apply them *[Req-157]*

RC-144: Only accept trajectory modifications from the controller assigned to a conflict *[Req-170]*

Process Model Parts & Required Feedback/Inputs

Feedback from aircraft: Received trajectory modifications *[Resp-1.3]*

Input from Resp-1.2: Controller assigned to resolve conflict *[RC-144]*

Input from Resp-1.5: Traffic priorities *[RC-131]*

Required Control Actions/Outputs

Control action to aircraft: Selected trajectory modifications *[RC-112]*

Figure F-3: Defined control elements for Resp-1.3

Resp-1.4: Ensure conformance with planned trajectory and any modifications *[Req-145]*

RC-27: If coordination was not effective, coordination is evaluated again to ensure that risks are adequately mitigated *[Req-50]*

RC-60: If a trajectory modification is not effective at resolving the collision, the reason for the modification not being effective must be determined so that an updated trajectory modification can account for it *[Req-84]*

RC-66: Re-evaluate an aircraft's trajectory if an aircraft deviates from its planned trajectory by more than <TBD> *[Req-91]*

RC-89: Account for any flight conditions that cause an aircraft to be unable to meet their expected navigational or maneuvering capabilities or trajectory constraints *[Req-115]*

RC-90: If prevailing flight conditions are affecting the navigational capabilities of multiple aircraft, verify those effects with other aircraft *[Req-116]*

RC-119: Ensure that aircraft needing trajectory modifications have received it, are executing it correctly and that the risk of collision or interference is no longer present *[Req-5]*

RC-134: Monitor an identified conflict until there is no longer a risk of collision *[Req-160]*

RC-142: Notify controllers assigned to a conflict if it remains unresolved *[Req-168]*

RC-145: Allow enough time for an assigned controller to resolve a conflict before flagging the conflict as unresolved *[Req-171]*

Process Model Parts & Required Feedback/Inputs

Feedback from the aircraft:

- Reason for trajectory deviation *[RC-60]*
- Prevailing flight conditions *[RC-89, RC-90]*

Feedback from Resp-1.1:

- Trajectory modifications *[RC-27, RC-119]*
- Identified conflicts *[RC-119]*

Input from Resp-1.2: Controller assigned to resolve conflict *[RC-145]*

Input from Resp-4: Consolidated airspace state *[RC-60]*

Internal Process Model Variable: Unresolved conflicts *[RC-66, RC-134]*

Required Control Actions/Outputs

Control actions to Resp-1.1:

- Unresolved collision risk *[RC-142]*
- Reason for ineffective traj. mod. *[RC-89]*

Control action to the aircraft: Request reason for ineffective trajectory modification *[RC-60]*

Figure F-4: Defined control elements for Resp-1.4

Resp-1.5: Set air traffic priorities to be enforced by the controller resolving a conflict *[Req-145]*

RC-7: Account for any users' constraints on mission execution in addition to access priorities to determine which impacts to operations are acceptable when coordinating aircraft *[Req-21]*

RC-44: Grant an aircraft experiencing an emergency the highest priority access to the airspace they need to address the emergency *[Req-68]*

RC-59: Ensure that the overall operational impact incurred by an aircraft is considered and minimized when making coordination decisions *[Req-78]*

RC-70: Ensure that any changes to relevant operational constraints are accounted for when issuing trajectory modifications *[Req-95]*

RC-96: Ensure that emergency response flights are given priority to carry out their missions *[Req-133]*

Process Model Parts & Required Feedback/Inputs

Feedback from Resp-1.1:

- Identified conflicts *[Resp-1.4]*
- Aircraft involved in a conflict *[Resp-1.4]*
- Ops constraints relevant for identified conflict *[RC-70]*

Input from Resp-4: Consolidated airspace state *[RC-7, RC-70, RC-96]*

Internal Process Model Variable: Accrued operational impact *[RC-59]*

Required Control Actions/Outputs

Control actions to Resp-1.1: Traffic priorities *[Resp-1.4]*

Figure F-5: Defined control elements for Resp-1.5

Resp-1.6: Establish when trajectory modification decisions need to be made centrally *[Req-124]*

RC-100: If NAS enters "centralized mode", it must have an associated end time when that mode ends *[Req-125]*

Process Model Parts & Required Feedback/Inputs

Feedback from Resp-1.2: Identified conflicts *[Resp-1.6]*

Input from Resp-4: Consolidated airspace state *[Resp-1.6]*

Internal Process Model Variables:

- Number of conflicts *[Resp-1.6]*
- Current and anticipated future traffic density *[Resp-1.6]*

Required Control Actions/Outputs

Control actions to Resp-1.2: Implement fully centralized collision avoidance *[Resp-1.6, RC-100]*

Figure F-6: Defined control elements for Resp-1.6

Appendix G Design Iteration 2 – STPA Analysis of Refined Conceptual Architecture

In Section 5.3.2, the refined conceptual architecture shown in Figure 39 was presented as though it was all created in a single iteration. In reality, however, several of the control actions and feedback shown in Figure 39 were only identified after the initial refined conceptual architecture was analyzed. This appendix presents the STPA analysis of the refined conceptual architecture to demonstrate how the results were used to identify both modifications to the initial conceptual architecture and the assignment constraints that were used to inform the creation of architecture options.

This appendix is organized as follows. First, the STPA results from analyzing the initial version of the refined conceptual architecture are shown. For causal scenarios that can be used to generate assignment constraints, the assignment constraints are highlighted in blue text at the end of the scenario. Then, a comparison of the initial and modified versions of the conceptual architecture are shown and the differences between them are highlighted and traced to the scenarios and requirements that were used to generate them. This demonstrates how the conceptual architecture can be modified based on some of the STPA results.

G.1 STPA Analysis of Initial Version of Refined Conceptual Architecture

For reference, the initial version of the refined conceptual architecture is shown in Figure G-1. There are some differences between this initial version and the modified version shown in Figure 39 in Section 5.3.2. These differences will be elaborated on later in Section G.2.

This STPA analysis is intentionally limited in scope because it is intended to only be a demonstration of how the initial version of the refined conceptual architecture would be analyzed. Thus, only the *Trajectory Modifications* control action provided from Resp-1.1 to the aircraft is analyzed and causal scenarios are identified for only a few UCCAs.

In addition, since this analysis is performed by updating the STPA analysis shown earlier in Appendix E, the UCCA tables are the same and will not be repeated in this appendix. Instead, only the UCCAs that were analyzed to generate scenarios will be shown along with the updated scenarios that were generated for them.

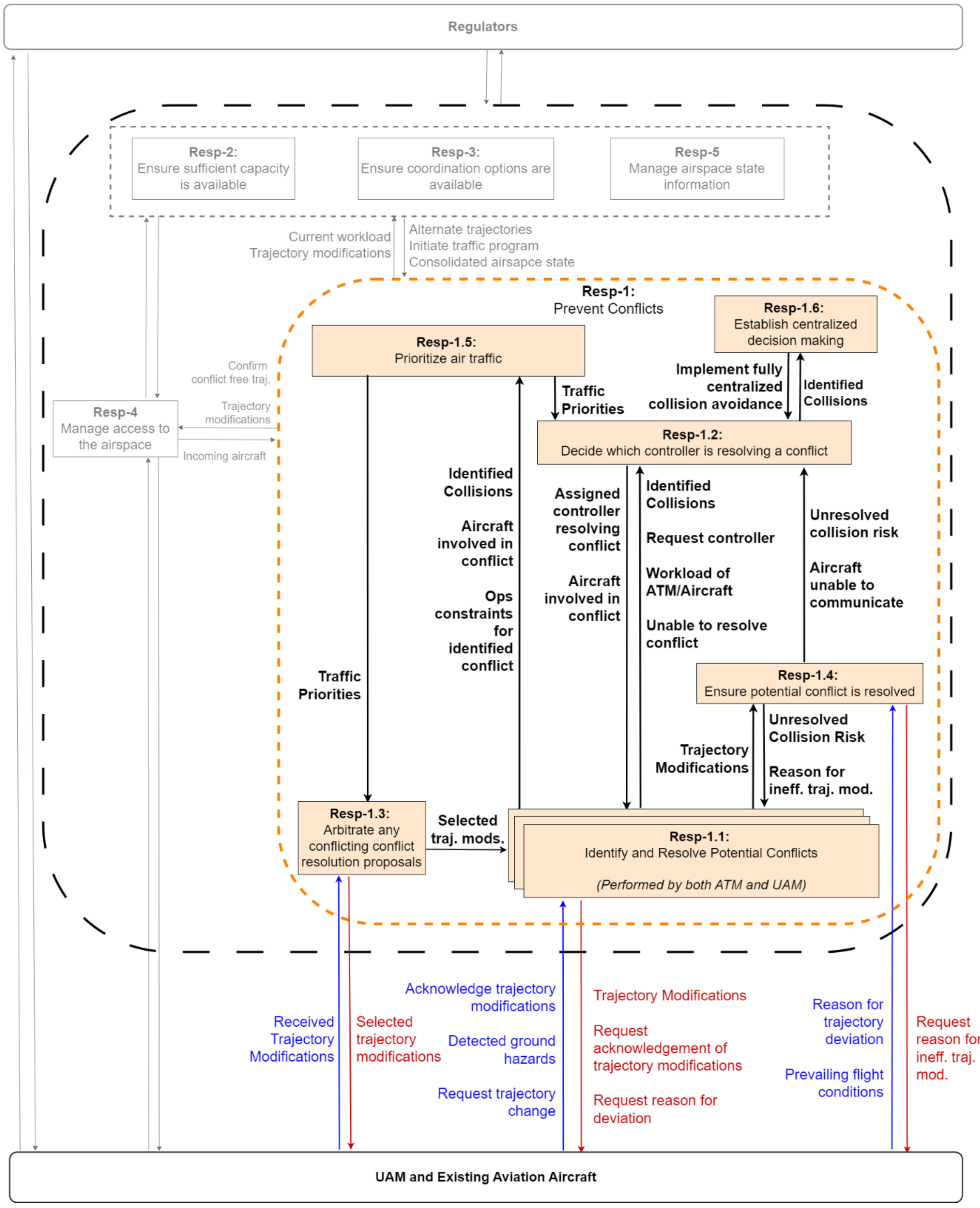

Regulators
Resp-2: Ensure sufficient capacity is available
Resp-3: Ensure coordination options are available
Resp-5 Manage airspace state information
Current workload
Trajectory modifications
Alternate trajectories
Initiate traffic program
Consolidated airsapce state
Resp-1: Prevent Conflicts
Resp-1.6: Establish centralized decision making
Resp-1.5: Prioritize air traffic
Confirm conflict free traj.
Trajectory modifications
Resp-4 Manage access to the airspace
Incoming aircraft
Traffic Priorities
Implement fully centralized collision avoidance
Identified Collisions
Resp-1.2: Decide which controller is resolving a conflict
Identified Collisions
Aircraft involved in conflict
Ops constraints for identified conflict
Assigned controller resolving conflict
Aircraft involved in conflict
Identified Collisions
Request controller
Workload of ATM/Aircraft
Unable to resolve conflict
Unresolved collision risk
Aircraft unable to communicate
Traffic Priorities
Resp-1.4: Ensure potential conflict is resolved
Trajectory Modifications
Unresolved Collision Risk
Reason for ineff. traj. mod.
Resp-1.3: Arbitrate any conflicting conflict resolution proposals
Selected traj. mods.
Resp-1.1: Identify and Resolve Potential Conflicts
(Performed by both ATM and UAM)
Received Trajectory Modifications
Selected trajectory modifications
Acknowledge trajectory modifications
Detected ground hazards
Request trajectory change
Trajectory Modifications
Request acknowledgement of trajectory modifications
Request reason for deviation
Reason for trajectory deviation
Prevailing flight conditions
Request reason for ineff. traj. mod.
UAM and Existing Aviation Aircraft

Figure G-1: Initial refined conceptual architecture

Scenarios for UCCA-1: Neither ATM nor aircraft provide trajectory modifications when the trajectories of two aircraft are in conflict [H-1]

CS-3.1.1. Neither ATM nor the aircraft are assigned to resolve the conflict and therefore no trajectory modifications are selected. This could occur if:

- **CS-3.1.1-1.** The conflict is correctly identified by either ATM or the aircraft and, as part of deciding who should resolve it, traffic priorities must be provided. However, if it takes too long to decide on traffic priorities, no one is assigned to resolve the conflict until traffic priorities are decided. *{Resp-1.5=ATM, Resp-1.5=Resp-1.2}*
- **CS-3.1.1-2.** The conflict is correctly identified but only by the aircraft. However, they are preoccupied with other tasks and are delayed in providing feedback about that conflict so that a decision can be made about who should resolve it. As a result, no one is assigned to resolve the collision. *[Req-161]*
- **CS-3.1.1-3.** Under a period of high workload, a conflict is wrongly perceived as not urgent and can be resolved later when the workload might be lower. However, if no better opportunity arises or workload prevents returning to the conflict to assign a controller, no one is ultimately assigned to resolve the conflict. In addition, that potential conflict is not monitored for resolution because a conflict is not monitored until trajectory modifications are issued. *[Req-160]*
- **CS-3.1.1-4.** The conflict is correctly identified and either ATM or the aircraft are assigned to resolve the conflict. However, if they do not process the control, they may not know that they have been assigned to resolve the conflict and therefore no one provides trajectory modifications to resolve it. *[Req-162]*
- **CS-3.1.1-5.** The aircraft identify an urgent conflict that needs to be resolved. However, they need to wait for a decision on who should resolve the conflict. By the time they receive that decision, there is not enough time to select trajectory modifications before the conflict occurs *{Resp-1.2 = Aircraft}*
- **CS-3.1.1-6.** The aircraft are assigned to resolve an urgent conflict that they did not identify. However, by the time they are notified and the aircraft synchronize their process models on what the conflict is and the aircraft that are involved, there is not enough time to select trajectory modifications before the conflict occurs. *{Resp-1.2 = Aircraft}*

CS-3.1.2. An inappropriate controller is assigned to resolve the collision. As a result, they are unable to select appropriate trajectories. This could occur if:

- **CS-3.1.2-1.** When the conflict was identified, the aircraft were assigned to resolve it. However, right after that decision is made, the system decides to switch to fully centralized collision avoidance. Thus, the conflict is re-assigned to ATM to resolve and the aircraft do not attempt to resolve it. However, this switch overwhelms ATM's capacity to resolve conflicts and it is unable to make a decision in time. *[Req-163]* *{Resp-1.2=ATM}*
- **CS-3.1.2-2.** ATM is assigned to resolve a conflict that could have been resolved by the aircraft based on inconsistent information about the current workload of ATM and the aircraft. Based on this incorrect information, it is wrongly believed that ATM's workload can handle resolving this conflict and the aircraft are too busy to resolve this conflict.

However, it is ATM that is too busy to resolve the conflict while the aircraft wait for trajectory modifications. *{Resp-1.2=ATM}*

CS-3.1.2-3. The aircraft are assigned to resolve a collision that initially does not involve a large number of aircraft. However, additional aircraft become identified as involved in the collision (e.g., more aircraft diverting to the same area for weather/aerodrome availability) and the conflict is simply updated to include these additional aircraft without re-evaluating the controller assigned to resolve the conflict. As a result, a large number of aircraft end up having to coordinate to select trajectory modifications instead of ATM resolving it centrally *[Req-164]*

CS-3.1.2-4. The aircraft are assigned to resolve an urgent conflict despite the fact that at least one of them is in a critical phase of flight where their workload is high. This assignment is made because of the urgency of the conflict but under out-of-date information about the context of the conflict (e.g., level of workload of that aircraft due to the critical phase of flight, trajectory constraints arising from traffic density) *(inadequate process model)*. As a result, the aircraft are unable to select appropriate trajectory modifications before a collision occurs. *{Resp-1.2=ATM}*

CS-3.1.3. The correct controller is assigned to resolve the conflict. However, no trajectory modifications are selected, or inadequate trajectory modifications are selected. This could occur because:

CS-3.1.3-1. The aircraft are correctly assigned to resolve a conflict. However, one of the aircraft is in a critical phase of flight (e.g., departure, final approach) and wrongly believes its trajectory has no room for deviation to avoid conflicting with aircraft that are about to depart. Thus, they rely on the other aircraft to modify their trajectories. However, if the airspace is constrained enough, the other aircraft may not be able to adequately modify their trajectories to prevent the conflict *[Req-165]*

CS-3.1.3-2. The aircraft are initially assigned to resolve a conflict but the conflict is re-assigned to ATM when the system decides to switch to fully centralized collision avoidance. However, the aircraft do not process this re-assignment and therefore ATM and the aircraft both select trajectory modifications that conflict. *[Req-166]*

CS-3.1.3-3. The aircraft are correctly assigned to resolve a conflict but do not have a consistent process model of each other's trajectory constraints. As a result, they propose trajectory modifications that are inappropriate for the other aircraft and ultimately are unable to select appropriate trajectory modifications before a collision occurs. *[Req-150]*

CS-3.1.3-4. The aircraft are correctly assigned to resolve a conflict. However, they select trajectory modifications that are inconsistent with the chosen traffic priorities (e.g., a higher priority aircraft is forced to deviate far off its original flight path to avoid a conflict). As a result, although the conflict is resolved, it results in some aircraft incurring unacceptable delays. *[Req-167]*

CS-3.1.4. The controller that is less equipped to resolve a conflict is correctly not assigned to resolve it. However, it does try to resolve the conflict anyway. This could occur if:

CS-3.1.4-1. The aircraft are assigned to resolve a conflict but they are unable to adequately resolve it. When this is discovered, the conflict is reassigned to ATM to resolve it. However, the aircraft are also notified that they did not resolve the conflict adequately and therefore start to re-resolve the conflict even though ATM is already assigned to resolve it. This results in duplicate trajectory modifications being issued. *[Req-168, Req-169]*

<u>**Scenarios for UCCA-11.1:**</u> ATM provides trajectory modifications (and the aircraft do not) when the modifications result in a collision

CS-3.11.1.1. ATM is inappropriately assigned to resolve a conflict and it chooses trajectory modifications that result in a collision. This could occur if:

CS-3.11.1.1-1. ATM is assigned to resolve the conflict because it was decided that all conflict resolution decisions should be made centrally. However, because ATM receives less timely feedback on flight conditions, it selects trajectory modifications that the aircraft cannot execute accurately enough in the current flight conditions and a collision occurs even though the aircraft were executing the trajectory modifications provided. *[Req-163]*

CS-3.11.1.1-2. ATM is assigned to resolve a conflict because out-of-date feedback about flight conditions or aircraft capabilities were used to make that decision. However, because flight conditions are changing often and ATM does not have as up-to-date feedback about flight conditions, it selects trajectory modifications that the aircraft cannot adequately execute, and a collision occurs. *{Resp-1.2=Aircraft}*

CS-3.11.1.1-3. ATM is assigned to resolve a conflict instead of the aircraft to avoid imposing additional workload onto the aircraft even though they are better suited to resolve the conflict *(inadequate control algorithm).* As a result, ATM struggles to select appropriate trajectory modifications. *{Resp-1.2=Aircraft}*

CS-3.11.1.2. ATM is correctly assigned to resolve a conflict and it chooses trajectory modifications that result in a collision. This could occur if:

CS-3.11.1.2-1. Near a busy aerodrome, ATM provides trajectory modifications to an aircraft to resolve a conflict based on an incorrect/out-of-date model of when aircraft are departing from the aerodrome. As a result, it provides trajectory modifications that conflict with the departure trajectory of an aircraft leaving the aerodrome and there is not enough time to resolve that conflict by the time it is identified *[Req-172]*

CS-3.11.1.2-2. The aircraft are assigned to resolve a conflict but an error in the communications channel results in some aircraft not receiving the full list of aircraft involved in the conflict. As a result, different aircraft have a different process model of which aircraft should be included in coordination efforts. As a result, some aircraft might be ignored even if they attempt to coordinate trajectory modifications because the other aircraft wrongly believe they are not part of the conflict being resolved *[Req-158]*

CS-3.11.1.2-3. In a previous conflict, ATM selected trajectory modifications that conflicted with those selected by the aircraft. While those conflicting trajectory modifications are being arbitrated, ATM selects trajectory modifications for this conflict without knowing what the result of the arbitration is. As a result, it selects trajectory modifications that conflict with the arbitrated trajectory modifications issued for the other conflict *[Req-177]*

CS-3.11.1.2-4. The aircraft are assigned to a conflict and begin coordinating trajectory modifications. However, some aircraft misinterpret the reference frame used by the other aircraft to specify trajectory constraints *(inadequate process model)*. As a result, some aircraft choose trajectory modifications that are actually in conflict with the other aircraft without realizing it. *[Req-178]*

<u>**Scenarios for UCCA-11.2:**</u> The aircraft provide trajectory modifications (and ATM does not) when the modifications result in a collision

CS-3.11.2.1. The aircraft are inappropriately assigned to resolve a conflict and they choose trajectory modifications that result in a collision. This could occur if:

CS-3.11.2.1-1. The aircraft are assigned to resolve a conflict involving an emergency response aircraft that is wrongly believed to have a known trajectory that won't change much. As a result, if their trajectory actually changes frequently or changes in a way that involves more aircraft, the aircraft can select trajectory modifications that contain conflicts. *[Req-164]*

CS-3.11.2.2. The aircraft are correctly assigned to resolve a conflict and they choose trajectory modifications that result in a collision. This could occur if:

CS-3.11.2.2-1. Two of the aircraft involved in the conflict select trajectory modifications that conflict with each other and assume that the other aircraft will select a different trajectory modification. By the time the conflicting trajectories are identified, there is not enough time to resolve them to avoid a collision. *[Req-173]*

CS-3.11.2.2-2. The aircraft are assigned to resolve a conflict. While selecting trajectory modifications, one of the aircraft begins executing its selected trajectory modification before it is confirmed that all aircraft involved in the conflict have selected adequate modifications. As a result, a collision occurs with other aircraft who have not yet selected appropriate trajectory modifications. *[Req-174]*

CS-3.11.2.2-3. Two aircraft are assigned the same priority and therefore they select very similar trajectory modifications at the same time. If they assume that the other aircraft will change its trajectory modification, they might both execute those similar trajectory modifications and cause a collision *[Req-174, Req-175]*

<u>**Scenarios for UCCA-17.1:**</u> ATM and the aircraft both provide trajectory modifications when they conflict with each other *[H-1]*

CS-3.17.1.1. Both UAM and the aircraft are assigned to resolve a conflict and they choose conflicting trajectory modifications. This might occur if:

CS-3.17.1.1-1. The aircraft are initially assigned to a conflict before it is realized that one of the aircraft is unable to perform self-separation (e.g., not equipped, not able to communicate). When this is realized, ATM is assigned to the conflict instead. However, if the aircraft do not process the re-assignment, they may end up selecting trajectory modifications while ATM is doing the same. *[Req-162, Req-166, Req-170]*

CS-3.17.1.1-2. The aircraft are assigned to resolve a conflict. However, before the aircraft have had enough time to resolve the conflict, premature feedback is received indicating that the conflict remains unresolved. As a result of this feedback, the conflict is re-assigned to ATM to resolve. If the aircraft do not process this re-assignment, they may end up selecting trajectory modifications while ATM is doing the same. *[Req-162, Req-166, Req-170, Req-171]* *{Resp-1.2 = Resp-1.4}*

CS-3.17.1.2. Only ATM or the aircraft are assigned to resolve a conflict but the other also attempts to resolve the conflict and they choose conflicting trajectory modifications. This could occur if:

CS-3.17.1.2-1. The aircraft are assigned to resolve a conflict but are unable to do so adequately. As a result, they are notified that the collision is unresolved. However, ATM also receives this notice and believes it should step in to help resolve it and does so even though it was not assigned to. *[Req-176]*

G.2 Identifying Required Modifications to the Initial Refined Conceptual Architecture

Using these causal scenarios, changes were made to the initial refined conceptual architecture shown in Figure G-1. Table G-1 lists the scenarios that were used to generate these changes, the requirement that was derived from those scenarios and the control actions or feedback that were added based on those requirements. Figure G-2 then shows the modified refined conceptual architecture. This is the same conceptual architecture that was shown in Figure 39 in Section 5.3.2, but with the added or removed control actions and feedback highlighted in green.

Table G-1: Scenarios and requirements that led to added control actions and feedback

Scenario ID	Requirement ID	Control Action/Feedback
CS-3.17.1.1-1	Req-170	Controller assigned to resolve conflict *(Control action from Resp-1.2 to Resp-1.3)*
CS-3.17.1.2-1	Req-176	Proposed assignment Transfer *(Control action from Resp-1.2 to Resp-1.1)*
		Assignment transfer accepted *(Feedback from Resp-1.1 to Resp-1.2)*
CS-3.1.1-4	Req-162	Acknowledge conflict to resolve *(Feedback from Resp-1.1 to Resp-1.2)*
CS-3.1.1-3	Req-160	Identified conflicts *(Feedback from Resp-1.1 to Resp-1.4)*
CS-3.17.1.1-2	Req-171	Controller assigned to resolve conflict *(Control action from Resp-1.2 to Resp-1.4)*
CS-3.1.4-1	Req-169	Persistent unresolved conflicts *(Feedback from Resp-1.4 to Resp-1.2)*

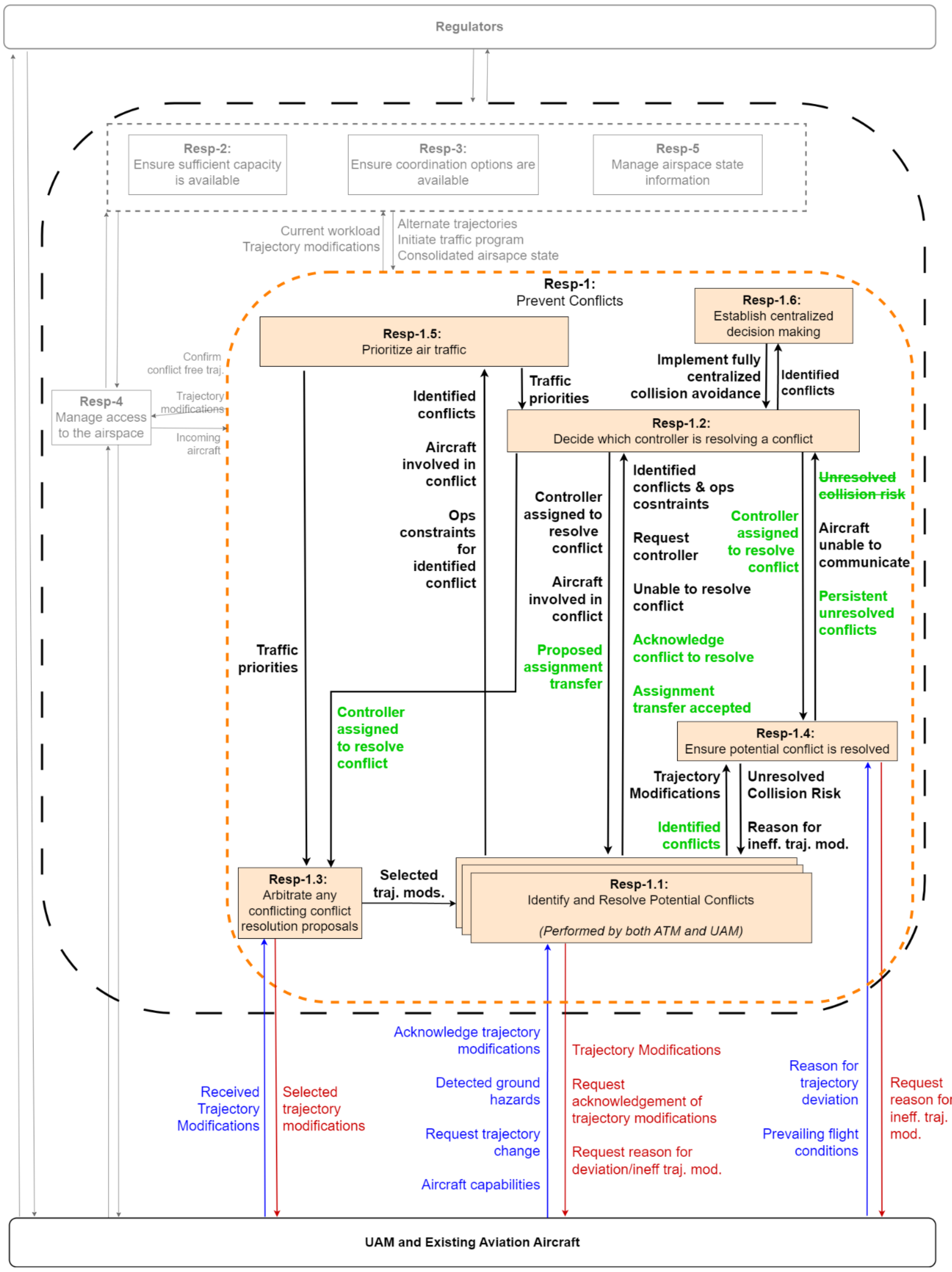

Regulators
Resp-2: Ensure sufficient capacity is available
Resp-3: Ensure coordination options are available
Resp-5 Manage airspace state information
Current workload
Trajectory modifications
Alternate trajectories
Initiate traffic program
Consolidated airsapce state
Resp-1: Prevent Conflicts
Resp-1.6: Establish centralized decision making
Resp-1.5: Prioritize air traffic
Confirm conflict free traj.
Resp-4 Manage access to the airspace
Trajectory modifications
Incoming aircraft
Identified conflicts
Traffic priorities
Implement fully centralized collision avoidance
Identified conflicts
Resp-1.2: Decide which controller is resolving a conflict
Aircraft involved in conflict
Ops constraints for identified conflict
Controller assigned to resolve conflict
Aircraft involved in conflict
Proposed assignment transfer
Identified conflicts & ops cosntraints
Request controller
Unable to resolve conflict
Acknowledge conflict to resolve
Assignment transfer accepted
Controller assigned to resolve conflict
~~Unresolved collision risk~~
Aircraft unable to communicate
Persistent unresolved conflicts
Traffic priorities
Controller assigned to resolve conflict
Resp-1.4: Ensure potential conflict is resolved
Trajectory Modifications
Identified conflicts
Unresolved Collision Risk
Reason for ineff. traj. mod.
Resp-1.3: Arbitrate any conflicting conflict resolution proposals
Selected traj. mods.
Resp-1.1: Identify and Resolve Potential Conflicts
(Performed by both ATM and UAM)
Received Trajectory Modifications
Selected trajectory modifications
Acknowledge trajectory modifications
Detected ground hazards
Request trajectory change
Aircraft capabilities
Trajectory Modifications
Request acknowledgement of trajectory modifications
Request reason for deviation/ineff traj. mod.
Reason for trajectory deviation
Prevailing flight conditions
Request reason for ineff. traj. mod.
UAM and Existing Aviation Aircraft

Figure G-2: Modified conceptual architecture with added control actions and feedback shown in green

Appendix H Design Iteration 2 – Analysis and Comparison of Architecture Options

This appendix shows the results from comparing the two architecture options in design iteration 2. The first option (A_4) is the ground-based conflict assignment architecture where Resp-1.2 is assigned to ATM. The second option (A_5) is the airborne conflict assignment architecture where Resp-1.2 is assigned to the aircraft. Table H-1 shows the full set of evaluation criteria that were identified and the comparison results for each architecture option. These evaluation criteria are sorted by type (e.g., decision making, control path). For each evaluation criterion, links are provided in square braces to the scenario(s) in Table H-2 used to generate them.

Table H-2 then presents the full architecture comparison table. This table contains (1) the scenarios used to compare the two architecture options, (2) the decisions about whether each scenario occurs for each architecture option, (3) any assumptions used to decide that a scenario does not occur for an architecture option, and (4) the evaluation criterion generated from that scenario. As in Appendix D, note that Table H-2 only includes scenarios where behavioral differences were observed.

Table H-1: Full set of evaluation criteria for comparison of architecture options A_4 and A_5

ID	Evaluation Criteria	Benefit (+) or Tradeoff (-)	
		A4	**A5**
	Decision Making Evaluation Criteria		
EC-2.1	**Responsiveness** of trajectory modification decisions when <u>the aircraft resolve an urgent conflict</u> *[Scenario 2.1]*	⊖	⊕
EC-2.2	**Stability** of decision about controller assigned to a conflict to prevent loss of separation when <u>waiting for controller to resolve conflict</u> *[Scenario 2.20]*	⊖	⊕
EC-2.3	**Capacity** to make conflict resolution decisions to prevent loss of separation when <u>selecting trajectory modifications</u> *[Scenario 2.25]*	⊖	⊕
EC-2.4	**Responsiveness** of trajectory modification decisions when <u>ATM resolves an urgent conflict</u> *[Scenario 2.2]*	⊕	⊖
EC-2.5	**Need to make** Conflict Assignment Transfer decisions when <u>an urgent conflict is identified</u> *[Scenario 2.5]*	⊕	⊖
EC-2.6	**Ability to make appropriate decisions** to accept/reject conflict assignments when <u>a controller is assigned a conflict to resolve</u> *[Scenario 2.22]*	⊕	⊖
EC-2.7	**Ease of coordinating** centralization and conflict assignment decisions when <u>switching to centralized decision making</u> *[Scenario 2.14]*	⊕	⊖
	Process Model Evaluation Criteria		
EC-2.8	**Ability to ensure adequate update of** controller assigned to conflict when <u>assigning two conflicts that occur close together in time</u> *[Scenario 2.4]*	⊖	⊕

ID	Evaluation Criteria	Benefit (+) or Tradeoff (-)	
		A4	A5
EC-2.9	**Available awareness** of aircraft workload when assigning conflicts to be resolved *[Scenarios 2.7, 2.8]*	⊖	⊕
EC-2.10	**Level of situational awareness** needed of aircraft involved in a conflict and relevant trajectory constraints when the aircraft resolve a conflict they did not identify *[Scenario 2.10]*	⊖	⊕
EC-2.11	**Required awareness** of aircraft or ATM workload to prevent loss of separation when assigning conflicts to be resolved *[Scenarios 2.17, 2.18]*	⊖	⊕
EC-2.12	**Ability to maintain alignment** of Controller Assigned to Conflict when receiving conflict assignment *[Scenarios 2.12, 2.13]*	⊖	⊕
EC-2.13	**Level of situational awareness** of aircraft involved in a conflict and relevant trajectory constraints when assigning conflicts to be resolved *[Scenario 2.11]*	⊕	⊖
EC-2.14	**Ability to maintain alignment** of Controller Assigned to Conflict when deciding who is resolving a conflict *[Scenarios 2.6, 2.23]*	⊕	⊖
EC-2.15	**Level of situational awareness** of future changes in airspace state to prevent loss of separation when resolving conflicts near aerodromes *[Scenario 2.26]*	⊕	⊖
EC-2.16	**Level of situational awareness** of trajectory constraints applicable for a conflict to prevent loss of separation when resolving an urgent conflict *[Scenario 2.27]*	⊕	⊖
	Feedback / External Inputs Evaluation Criteria		
EC-2.17	**Timeliness** of flight conditions and aircraft capabilities feedback when assigning conflicts to be resolved *[Scenarios 2.9, 2.15]*	⊖	⊕
EC-2.18	**Ability to process** identified conflicts inputs when the workload of the controller processing that feedback is high *[Scenario 2.3]*	⊖	⊕
EC-2.19	**Timeliness** of feedback about aircraft arrivals and departures to prevent loss of separation when resolving conflicts near aerodromes *[Scenario 2.19]*	⊖	⊕
EC-2.20	**Ability to evaluate and verify** aircraft to be included in conflict resolution to prevent loss of separation when selecting trajectory modifications *[Scenario 2.24]*	⊖	⊕
EC-2.21	**Ability to evaluate and verify** requests to resolve a conflict to prevent loss of separation when ATM or the aircraft request to resolve a conflict *[Scenario 2.16]*	⊕	⊖
EC-2.22	**Ability to respond appropriately** to centralization inputs to prevent loss of separation when assigning conflicts to be resolved *[Scenario 2.21]*	⊕	⊖

Table H-2: Comparison results for the centralized (A_1) and decentralized (A_2) collision avoidance architecture options

ID	**Scenario**	**Scenario Occurs?**	**Evaluation Criteria**
2.1	The aircraft identify an urgent conflict that needs to be resolved. However, <controller performing Resp-1.2> takes too long to decide who should resolve a conflict. By the time the aircraft receive that decision, there is not enough time to select trajectory modifications before the conflict occurs	**A_4: Yes** **A_5: No** *Assumption: Since the aircraft identify a conflict and are also the ones deciding who should resolve it, they can respond to an urgent conflict quicker*	**EC-2.1: Responsiveness** of trajectory modification decisions to prevent loss of separation when the aircraft resolves an urgent conflict
2.2	ATM identifies an urgent conflict that needs to be resolved. However, <controller performing Resp-1.2> takes too long to decide who should resolve a conflict (inadequate control algorithm). By the time ATM receives that decision, there is not enough time to select trajectory modifications before the conflict occurs	**A_4: No** *Assumption: If ATM identifies a conflict and it also decides who should resolve it, it can respond to an urgent conflict quicker* **A_5: Yes**	**EC-2.4: Responsiveness** of trajectory modification decisions to prevent loss of separation when ATM resolves an urgent conflict
2.3	<Controller not performing Resp-1.2> identifies a conflict and provides that feedback, but <controller performing Resp-1.2> does not process that feedback appropriately because it is experiencing high workload. As a result, <controller not performing Resp-1.2> wrongly believe <controller performing Resp-1.2> is aware of the conflict but it is not. The conflict therefore remains unresolved by <controller performing Resp-1.2>.	**A_4: Yes** **A_5: No** *Assumption: Even if one of the aircraft is experiencing a high workload and does not process the indication from ATM, other aircraft might not be and could process the indication and decide to resolve the conflict (or not)*	**EC-2.18: Ability to process** identified conflicts inputs to prevent loss of separation when the workload of the controller processing that feedback is high

ID	Scenario	Scenario Occurs?	Evaluation Criteria
2.4	Two conflicts occur close together and <controller performing Resp-1.2> wrongly believe they already requested <controller not performing Resp-1.2> to resolve both conflicts even though that assignment was for the earlier conflict, not the more recent one. <Controller performing Resp-1.2> therefore does not issue a new assignment for the more recent conflict and that conflict goes unresolved.	**A_4: Yes** **A_5: No** *Assumption: The aircraft are monitoring their own trajectories closely and so would not forget to either resolve a conflict themselves or have ATM take over to resolve it*	**EC-2.8: Ability to ensure adequate update of** controller assigned to conflict when assigning two conflicts that occur close together in time
2.5	<controller performing Resp-1.2> identifies an urgent conflict and initially believe they can resolve the conflict but realize they are unable to. Because of the urgency of the conflict, by the time they try to request <controller not performing Resp-1.2> to take over, there is not enough time for it to resolve it before a collision occurs.	**A_4: No** *Assumption: ATM, with its broader situational awareness, would be able to resolve an urgent conflict if it was needed to (and it was the one who identified it)* **A_5: Yes**	**EC-2.5: Need to make** Conflict Assignment Transfer decisions to prevent loss of separation when an urgent conflict is identified
2.6	<Controller performing Resp-1.2> identifies a conflict but has conflicting process models about who is resolving the conflict. As a result, <controller performing Resp-1.2> does not assign a controller to resolve the conflict and does not resolve the conflict itself	**A_4: No** *Assumption: Within ATM, its process model will always be consistent about whether it is resolving a conflict itself or assigning it to the aircraft.* **A_5: Yes**	**EC-2.14: Ability to maintain alignment** of Controller Assigned to Conflict to prevent loss of separation when deciding who is resolving a conflict

ID	Scenario	Scenario Occurs?	Evaluation Criteria
2.7	<Controller performing Resp-1.2> attempts to resolve a conflict that should be assigned to <controller not performing Resp-1.2> because <Controller performing Resp-1.2> has incorrect information about the current workload of <controller not performing Resp-1.2>. However, <Controller performing Resp-1.2> is itself experiencing high workload and cannot select appropriate trajectory modifications before a collision occurs.	**A_4: Yes** **A_5: No** *Assumption: The aircraft would not try to account for ATM's workload and just make the request for ATM to help resolve a conflict if needed*	**EC-2.9: Available awareness** of aircraft workload to prevent loss of separation when assigning conflicts to be resolved
2.8	<Controller performing Resp-1.2> assigns the aircraft to resolve a collision that initially does not involve a large number of aircraft. However, additional aircraft become identified as involved in the conflict. Although <controller performing Resp-1.2> is aware that the number of aircraft in the conflict is growing, it keeps the conflict assigned to the aircraft. As a result, a large number of aircraft end up having to coordinate to select trajectory modifications.	**A_4: Yes** **A_5: No** *Assumption: Once the aircraft believe they can't resolve the conflict adequately, they will request ATM's assistance as soon as possible*	**EC-2.9: Available awareness** of aircraft workload to prevent loss of separation when assigning conflicts to be resolved
2.9	<Controller performing Resp-1.2> assigns the aircraft to resolve an urgent conflict even though at least one of them is in a critical phase of flight where their workload is high. This assignment is made because of the urgency of the conflict but under inaccurate information about the capabilities/operational constraints of that aircraft (e.g., limitations on low altitude maneuvering). As a result, the aircraft are unable to select appropriate trajectory modifications before a collision occurs.	**A_4: Yes** **A_5: No** *Assumption: Since the aircraft exchange trajectory constraints and are directly gathering weather feedback, they have more timely access to this information than ATM does*	**EC-2.17: Timeliness** of flight conditions and aircraft capabilities feedback to prevent loss of separation when assigning conflicts to be resolved

ID	Scenario	Scenario Occurs?	Evaluation Criteria
2.10	<Controller performing Resp-1.2> assigns the aircraft to resolve an urgent conflict that they did not identify based on the urgency of the conflict. However, by the time the aircraft synchronize their process models on what the conflict is and the aircraft that are involved, there is not enough time to select trajectory modifications before the conflict occurs.	**A_4: Yes** **A_5: No** *Assumption: When the aircraft perform Resp-1.2, they have greater situational awareness of the airspace and will be faster in synchronizing their process model of the conflict identified by ATM.*	**EC-2.10: Level of situational awareness** needed of aircraft involved in a conflict and relevant trajectory constraints to prevent loss of separation when the aircraft resolve a conflict they did not identify
2.11	<Controller performing Resp-1.2> identifies a conflict but does not adequately process feedback on the relevant operational or trajectory constraints for that conflict. Thus, they correctly assign the aircraft to resolve the conflict but wrongly omit some aircraft from the list of aircraft involved in the conflict. As a result, only a subset of the aircraft coordinate to resolve the conflict.	**A_4: No** *Assumption: ATM would have broader awareness needed to accurately determine the operational or trajectory constraints relevant for a conflict* **A_5: Yes**	**EC-2.13: Level of situational awareness** of aircraft involved in a conflict and relevant trajectory constraints to prevent loss of separation when assigning conflicts to be resolved
2.12	<Controller performing Resp-1.2> correctly assigns the aircraft to resolve a conflict. However, delays in when that assignment is received by the various aircraft result in delays in the aircraft beginning to coordinate trajectory modifications (inadequate control path). As a result of these delays, the conflict is not adequately resolved before a collision occurs.	**A_4: Yes** **A_5: No** *Assumption: With the aircraft coordinating on who should resolve a conflict, a delay in starting to select trajectory modifications would not occur*	**EC-2.12: Ability to maintain alignment** of Controller Assigned to Conflict when receiving conflict assignment

ID	Scenario	Scenario Occurs?	Evaluation Criteria
2.13	<Controller performing Resp-1.2> assigns the aircraft to resolve a conflict. Based on the provided traffic priorities, the aircraft wait for the highest priority aircraft to select its trajectory modifications. However, if the highest priority aircraft is delayed in recognizing that it needs to resolve a conflict, that will delay all other aircraft in selecting trajectory modifications as well	**A_4: Yes** **A_5: No** *Assumption: If the aircraft have identified the conflict, they know they need to resolve it even if they are still waiting on traffic priorities to be provided by ATM*	**EC-2.12: Ability to maintain alignment** of Controller Assigned to Conflict when <u>receiving conflict assignment</u>
2.14	Either ATM or the aircraft identify a conflict and <controller performing Resp-1.2> decides to resolve it. However, while they are resolving the conflict, ATM also indicates that it is implementing centralized collision avoidance. Based on this input, <controller performing Resp-1.2> attempts to transfer the conflict to ATM. If ATM is unable to resolve the conflict (e.g., too little time remaining to collision), the conflict remains unresolved.	**A_4: No** *Assumption: In this architecture, ATM retains sole decision-making authority over assignment of conflicts to aircraft. Thus, the control action to switch to centralized collision avoidance "mode" is internal to ATM and not known to the aircraft* **A_5: Yes**	**EC-2.7: Ease of coordinating** centralization and conflict assignment decisions to prevent loss of separation when <u>switching to centralized decision making</u>
2.15	<Controller performing Resp-1.2> decides to resolve a conflict based on out-of-date feedback about flight conditions or aircraft capabilities. However, because flight conditions are changing often and <Controller performing Resp-1.2> does not receive timely feedback about flight conditions, it selects trajectory modifications that the aircraft cannot adequately execute, and a collision occurs.	**A_4: Yes** **A_5: No** *Assumption: The aircraft will have more timely feedback about flight conditions and aircraft capabilities than ATM and therefore could make more appropriate resolution decisions when those factors are important to consider*	**EC-2.17: Timeliness** of flight conditions and aircraft capabilities feedback to prevent loss of separation when <u>assigning conflicts to be resolved</u>

ID	Scenario	Scenario Occurs?	Evaluation Criteria
2.16	<Controller not performing Resp-1.2> indicates a conflict to <controller performing <Resp-1.2> and requests that it be allowed to resolve the conflict. Even though <controller not performing Reps-1.2. is not able to adequately resolve the conflict, <controller performing Resp-1.2> complies with the request. As a result, <controller not performing Resp-1.2> selects trajectory modifications that result in a collision	**A_4: No** *Assumption: ATM verifies any request from the aircraft to resolve a conflict. Given ATM's broader situational awareness of what is happening in the airspace, it is assumed that ATM would know if the aircraft can or cannot adequately resolve a conflict* **A_5: Yes**	**EC-2.21: Ability to evaluate and verify** requests to resolve a conflict to prevent loss of separation when ATM or the aircraft request to resolve a conflict
2.17	<Controller resolving Resp-1.2> decides to resolve a conflict itself to avoid imposing additional workload on <controller not performing Resp-1.2> even though <controller not performing Resp-1.2> is better suited to resolve the conflict. However, <controller performing Resp-1.2> struggles to select appropriate trajectory modifications due to workload or other conditions and selects modifications that result in a conflict	**A_4: Yes** **A_5: No** *Assumption: Once the aircraft believe they can't resolve the conflict adequately, they will request ATM's assistance and do not need to consider ATM's workload because ATM can mitigate its workload in other ways*	**EC-2.11: Required awareness** of aircraft or ATM workload to prevent loss of separation when assigning conflicts to be resolved
2.18	<Controller resolving Resp-1.2> decides to assign the conflict to <controller not performing Resp-1.2> even though <controller performing Resp-1.2> is better suited to resolve the conflict because <controller performing Resp-1.2> is experiencing high workload. However, <controller not performing Resp-1.2> struggles to select appropriate trajectory modifications due to workload or other conditions and selects modifications that result in a conflict	**A_4: Yes** **A_5: No** *Assumption: Once the aircraft believe they can't resolve the conflict adequately, they will request ATM's assistance and do not need to consider ATM's workload because ATM can mitigate its workload in other ways*	**EC-2.11: Required awareness** of aircraft or ATM workload to prevent loss of separation when assigning conflicts to be resolved

ID	Scenario	Scenario Occurs?	Evaluation Criteria
2.19	Near a busy aerodrome, <controller performing Resp-1.2> decides to assign ATM to resolve a conflict because it wrongly believing that the airspace near that aerodrome is too busy for the aircraft to resolve the conflict themselves. However, <controller performing Resp-1.2> provides trajectory modifications to the aircraft to resolve a conflict based on an out-of-date model of when aircraft are departing from the aerodrome. As a result, it provides trajectory modifications that conflict with the departure trajectory of an aircraft leaving the aerodrome	**A_4: Yes** **A_5: No** *Assumption: The aircraft could receive more up-to-date information about aerodrome departures by talking directly to other aircraft than for ATM to put that information together.*	**EC-2.19: Timeliness** of feedback about aircraft arrivals and departures to prevent loss of separation when <u>resolving conflicts near aerodromes</u>
2.20	The <controller performing Resp-1.2> initially assigns the conflict to <controller not performing Resp-1.2> to resolve, but then decides that they are taking too long to resolve the conflict. <Controller performing Resp-1.2> therefore reassigns the conflict to itself to resolve even though <controller not performing Resp-1.2> was almost done resolving a conflict. However, there is not enough time to adequately resolve the conflict.	**A_4: Yes** **A_5: No** *Assumption: Because the aircraft cannot unilaterally direct ATM to take over resolving a conflict, they are less likely to repeatedly change their decision of who should resolve a conflict*	**EC-2.2: Stability** of decision about controller assigned to a conflict to prevent loss of separation when <u>waiting for controller to resolve conflict</u>
2.21	ATM indicates that it is implementing centralized collision avoidance. However, <controller performing Resp-1.2> receives this indication and assumes it should transfer all conflicts they are resolving to ATM to resolve (inadequate control algorithm). This therefore overwhelms ATM and it selects inadequate trajectory modifications that result in a collision.	**A_4: No** *Assumption: In this architecture, ATM retains sole decision making authority over assignment of conflicts to aircraft. Thus, the control action to switch to centralized collision avoidance "mode" is internal to ATM and not known to the aircraft* **A_5: Yes**	**EC-2.22: Ability to respond appropriately** to centralization inputs to prevent loss of separation when <u>assigning conflicts to be resolved</u>

ID	Scenario	Scenario Occurs?	Evaluation Criteria
2.22	<Controller performing Resp-1.2> requests <controller not performing Resp-1.2> to resolve a conflict even though <controller performing Resp-1.2> is better equipped to resolve it. Because <controller not performing Resp-1.2> wrongly believes it must accept the request, it selects modifications that result in a collision.	**A_4: No** *Assumption: The aircraft will always indicate to ATM if they cannot resolve a conflict and will not feel "obligated" to accept a request* **A_5: Yes**	**EC-2.6: Ability to make appropriate decisions** to accept/reject conflict assignments when a controller is assigned a conflict to resolve
2.23	<Controller performing Resp-1.2> identify a conflict but it becomes misaligned about who is resolving the conflict. As a result, <Controller performing Resp-1.2> provides conflicting feedback both requesting the assistance of <controller not performing Resp-1.2> and indicating that <controller performing Resp-1.2> is resolving the conflict itself. When <controller performing Resp-1.2> receives this conflicting feedback, it decides it should just resolve the conflict even though it is not able to do so adequately	**A_4: No** *Assumption: Because ATM is the sole decision maker for Resp-1.2 in this scenario, ATM will maintain a consistent process model of who is assigned to resolve a conflict* **A_5: Yes**	**EC-2.14: Ability to maintain alignment** of Controller Assigned to Conflict to prevent loss of separation when deciding who is resolving a conflict
2.24	<Controller performing Resp-1.2> assigns the conflict to <controller not performing Resp-1.2> to resolve but inadvertently leaves out several aircraft that should be included in coordination to prevent the conflict (inadequate control algorithm). As a result, <controller not performing Resp-1.2> does not adequately coordinate their trajectory modifications and some result in a collision.	**A_4: Yes** **A_5: No** *Assumption: Because of ATM's other responsibilities, even if the aircraft inadvertently omit some aircraft from the set of aircraft to be included in coordination, ATM will recognize and correct for that*	**EC-2.20: Ability to evaluate and verify** aircraft to be included in conflict resolution to prevent loss of separation when selecting trajectory modifications

ID	Scenario	Scenario Occurs?	Evaluation Criteria
2.25	The aircraft receive a <conflict assignment/request to resolve conflict> from ATM. However, the aircraft are in a high workload situation or resolving other conflicts and therefore does not immediately begin resolving the conflict. As a result, they are unable to adequately coordinate to prevent the conflict and they choose trajectory modifications that result in a collision.	**A_4: Yes** **A_5: No** *Assumption: The aircraft are monitoring their own trajectories closely and so would not forget to either resolve a conflict themselves or have ATM take over to resolve it*	**EC-2.3: Capacity** to make conflict resolution decisions to prevent loss of separation when <u>selecting trajectory modifications</u>
2.26	<Controller performing Resp-1.2> decides to assign the aircraft to resolve a conflict, believing that the aircraft are receiving more up-to-date information about departure and arrival trajectories from the aerodrome. However, due to an event occurring at another aerodrome (e.g., weather), numerous aircraft are about to divert to this aerodrome. As the traffic density increases, the aircraft are unable to select appropriate trajectory modifications and there is not enough time for ATM to assist before a collision occurs.	**A_4: No** *Assumption: With ATM's broader situational awareness, it would know to transfer the conflict back to itself if necessary when the airspace density changes as it does in this scenario* **A_5: Yes**	**EC-2.15: Level of situational awareness** of future changes in airspace state to prevent loss of separation when <u>resolving conflicts near aerodromes</u>
2.27	<Controller performing Resp-1.2> decides to assign the aircraft to resolve an urgent conflict even though their workload is high. This assignment is made because of the urgency of the conflict but under inaccurate information about all the applicable trajectory constraints or while lacking information about the anticipated future state of the airspace. As a result, the aircraft are unable to select appropriate trajectory modifications before a collision occurs.	**A_4: No** *Assumption: ATM will have broader situational awareness to identify all trajectory constraints for a conflict* **A_5: Yes**	**EC-2.16: Level of situational awareness** of trajectory constraints applicable for a conflict to prevent loss of separation when <u>resolving an urgent conflict</u>