A Top-Down, Safety-Driven Approach to Architecture Development for Complex Systems

by

Justin Wei Siang Poh

B.S., Olin College of Engineering (2016)

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

MASTER OF SCIENCE IN AERONAUTICS AND ASTRONAUTICS
AT THE
MASSACHUSETTS INSTITUTE OF TECHNOLOGY

FEBRUARY 2022

© 2022 Massachusetts Institute of Technology. All rights reserved.

Signature of Author: _____
Department of Aeronautics and Astronautics
January 21, 2022

Certified by: _____
Nancy G. Leveson, Ph.D.
Professor of Aeronautics and Astronautics
Thesis Supervisor

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

A Top-Down, Safety-Driven Approach to Architecture Development for Complex Systems

by

Justin Wei Siang Poh

Submitted to the Department of Aeronautics and Astronautics
on January 21, 2022 in partial fulfillment of the
requirements for the degree of
Master of Science in Aeronautics and Astronautics

Abstract

Architecture development is an important part of the systems engineering process because the system architecture forms the foundation on which the rest of the system design is based. In addition, the system architecture plays a key role in determining the behavior of the system and represents a set of design decisions made to solve a design problem. Because modern systems are increasingly complex and software-intensive, they require architectures that fully consider system-level interactions and unsafe behaviors and ensures that the responsibilities necessary to ensure safety are carried out effectively. Furthermore, the architecture development process should organize design information in a way that assists system designers and reviewers with managing system complexity and developing an understanding of the system design and its underlying rationale.

This thesis proposes a new top-down, safety-driven approach to architecture development that is based on systems theory and incorporates a hazard analysis at the beginning of the design process to drive the identification of system-level requirements. This approach ensures that the system and its environment are analyzed as a whole and emergent properties such as safety are considered as early as possible. Using a structured process and appropriate types of abstraction, this new approach to architecture development facilitates obtaining more information about how the system needs to behave before creating a series of candidate architecture options and assessing the tradeoffs between them.

The proposed approach is applied to create a conceptual architecture for a human pilot and automated flight controller performing medevac flights in Degraded Visual Environments (DVEs). This example illustrates how the new approach can be used to develop architectures in a top-down, safety-driven manner and shows how the design information obtained using this new approach can be used to make more informed architectural decisions.

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

Acknowledgements

This thesis represents a milestone in a professional journey for me that started when I began my first job after graduating from Olin College of Engineering. While I do not know where this journey will take me next, I wanted to use this opportunity to acknowledge the contributions of those who have taught me so much and express my heartfelt thanks for everything they have done for me.

I want to start by thanking Motional, where I began my first job out of college. Over the four years I worked for Motional, the company gave me many invaluable opportunities to experience the difficulties and challenges of creating system architectures for automated vehicles. I was also fortunate to be able to work with many extremely talented colleagues and managers from whom I learned a great deal about all the various aspects that an architecture must account for. It is because of these engineering experiences I had at Motional that I discovered my interest in systems engineering practice and acquired a passion to do my best to improve it. I am extremely grateful that I had the opportunity to work for Motional and I owe a debt of gratitude to all my former colleagues and managers.

I would also like to thank my thesis supervisor, Professor Nancy Leveson, for teaching me so much over the last 18 months. Not only did she show me new ways to think about system safety, system engineering and system specifications but she also taught me so much about how to communicate those ideas effectively in a variety of different settings. I have never had an advisor or professor who has given me such detailed feedback on my writing and presentations as she has. I am very grateful to have met Nancy and had the opportunity to learn from her.

Next, I want to thank my research group for not only their friendship but also the help they have given me during my time at MIT thus far. To Sam Yoo, Dro Gregorian and Andrew Kopeikin, thank you for teaching me so much about how fixed wing and rotary wing aircraft are flown. Given my lack of experience with aviation in general and my lack of experience as a pilot, I could not have done this research without your help. To Lawrence Wong, Adam Munekata and Alex Hillman, thank you for all the feedback you've given me in my numerous presentations of my research to our group. I also must thank Dr John Thomas who has offered extremely valuable advice and feedback throughout my time at MIT not only with regards to my thesis but also with regards to STPA in general. In fact, it was because of a presentation John gave at Motional several years ago that I became interested in STPA and system safety. So, thank you for all your help and your enthusiastic teaching style.

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 both before and during my time at MIT. This journey would not have been possible without you and I will be forever thankful for you.

Table of Contents

Abstract 3

Acknowledgements 5

Table of Figures 9

Table of Tables 10

Chapter 1 Introduction 12

1.1 The Role of System Architecture in Systems Engineering 12

1.2 Limitations of Current Methods for Developing System Architectures 13

1.3 Research Objectives and Approach 15

Chapter 2 Background 16

2.1 Current Methods for Developing System Architectures 16

2.1.1 Overview of Current Methods for Developing System Architectures 16

2.1.2 Shortcomings of Current Methods 18

2.2 Systems Theory and Emergent Properties 20

2.3 Management of Complexity and Hierarchical Abstraction 21

2.4 STAMP and STPA 22

2.5 Using STPA to Generate a Conceptual Architecture 28

2.6 Intent Specifications 28

Chapter 3 A New Method for Top-Down, Safety-Driven Architecture Development 31

3.1 Overview of the Method 31

3.2 Step 1: Analyze the System and Its Environment 31

3.3 Step 2: Define Solution-Neutral System Requirements 32

3.4 Step 3: Define System-Level Behavior 32

3.5 Step 4: Create and Assess Architecture Options 34

Chapter 4 Case Study: Creating a Conceptual Architecture for Performing Medevac Flights in Degraded Visual Environments (DVEs) 38

4.1 Case Study Overview 38

4.2 Step 1: Analyzing the System and Its Environment 39

4.2.1 Defining Purpose of the Analysis, System Boundary and System Goal 39

4.2.2 Initial Assumptions and Constraints 40

4.2.3 STPA Step 1: Losses, System-Level Hazards & System Constraints 41

4.2.4 STPA Step 2: Safety Control Structure 41

4.2.5 STPA Step 3: Unsafe Control Actions (UCAs) 44

4.2.6 STPA Step 4: Identify Loss Scenarios 45

4.3 Step 2: Defining Solution-Neutral, System-Level Requirements 48

4.4 Step 3: Defining System-Level Behavior 51
4.4.1 Defining Responsibilities 52
4.4.2 Defining Decision-Making Strategies, Refined Control Actions and Required Process Model Parts 54
4.4.3 Defining Sources of Feedback 55
4.4.4 Defining Timing Requirements 57

4.5 Step 4, Part 1: Creating and Assessing Architecture Options for the Overall System 60
4.5.1 Overview of Architecture Options 60
4.5.2 Architecture Option 1: The Conventional Flight Operations System 61
4.5.3 Architecture Analysis of Option 1 69
4.5.4 Architecture Option 2: The Enhanced Flight Operations System 73
4.5.5 Architecture Analysis of Option 2 78
4.5.6 Comparison of Architecture Options 81

4.6 Step 4, Part 2: Creating and Assessing Architecture Options for the Piloting Controller 85
4.6.1 Overview of Architecture Options 85
4.6.2 Architecture Option 1: Human-Piloted Aircraft with ASEC as a Decision Aid 86
4.6.3 Architecture Analysis of Option 1 94
4.6.4 Architecture Option 2: ASEC Proposes Flight Trajectories for Human Pilot to Execute 99
4.6.5 Architecture Analysis of Option 2 105
4.6.6 Architecture Option 3: Human Pilot Supervises Automated ASEC's Control of Flight 110
4.6.7 Architecture Analysis of Option 3 114
4.6.8 Comparison of Architecture Options 122

Chapter 5 Conclusions 126

5.1 Summary 126

5.2 Future Work 127

Chapter 6 References 129

Appendix A UCAs Identified For STPA Analysis of FOS 131

Appendix B Loss Scenarios Identified For STPA Analysis of FOS 133

Appendix C System-Level Requirements 139

Appendix D FOS System-Level Behavior Information 144

Appendix E Responsibility Assignments for Part 1 Architecture Option 1 166

Appendix F Responsibility Assignments for Part 2 Architecture Option 1 173

Appendix G Additional STPA Analysis for Architecture Option 3 179

Table of Figures

Figure 1: The Systems Engineering Vee model (adapted from [1] - [3]) 12
Figure 2: The three types of systems defined by degree of randomness and complexity (from [8]) 20
Figure 3: Illustration of means-ends abstraction with examples (adapted from [11]) 22
Figure 4: A controller enforcing controls on a system's behavior and the interactions between its components (from [14]) 24
Figure 5: The control structures for a system during system development (left side) and once the system is operational (right side) (from [14]) 25
Figure 6: Illustration of a simple control loop involving a controller with a control algorithm and a process model (from [14]) 26
Figure 7: The STPA Process (from [14]) 27
Figure 8: Illustration of how STPA identifies UCAs and the loss scenarios based on flaws in the four main parts of a control loop (from [14]) 27
Figure 9: An Illustration of the form of an Intent Specification (from [19]) 29
Figure 10: Overview of the new safety-driven approach to concept and architecture development 31
Figure 11: Illustration of the five parts of system-level behavior in blue 33
Figure 12: The order in which system-level behavior information should be defined 34
Figure 13: The 2-part architecture creation process and its inputs and outputs 34
Figure 14: Generic illustration of Part 1 of the Architecture Creation Process 36
Figure 15: Generic illustration of Part 2 of the Architecture Creation Process 37
Figure 16: Control Structure with Abstracted Piloting Controller Highlighted 42
Figure 17: Example 1 of how solution-neutral, system-level requirements are generated 48
Figure 18: Example 2 of how solution-neutral, system-level requirements are generated 49
Figure 19: Part 1 Architecture Option 1 Control Structure 68
Figure 20: Performance Requirements for Each Part of the System 72
Figure 21: Part 1 Architecture Option 2 Control Structure 77
Figure 22: Part 2 Architecture Option 1 Control Structure 93
Figure 23: Part 2 Architecture Option 2 Control Structure 104
Figure 24: Part 2 Architecture Option 3 Control Structure 113
Figure 25: Option 3 control structure with new control actions from human pilot to ASEC circled in red 115

Table of Tables

Table 1: Description of Components in the FOS 39
Table 2: Example UCAs for Actuator Movements Control Action 44
Table 3: Example Loss Scenarios for UCA-1.2 45
Table 4: Example Loss Scenarios for UCA-1.5 47
Table 5: System-Level Requirements List 50
Table 6: Descriptions of the Five Aspects of System-Level Behavior Information 51
Table 7: Translating system-level requirements into responsibilities 52
Table 8: Decision-Making Strategy, Refined Control Actions and Process Model Parts for SR-18 and SR-22 54
Table 9: Feedback Information for Current Navigation State and Current Airspace State 56
Table 10: Example Timing Requirements for Non-Feedback-Validation Responsibilities 58
Table 11: Example Timing Requirements for Feedback Validation Responsibilities 59
Table 12: Assignments of Process Model Parts for Part 1 Architecture Option 1 61
Table 13: Flight-Related Non-Feedback-Validation Responsibilities Assigned Only to the PC in Part 1 Architecture Option 1 63
Table 14: Part 1 Architecture Option 1 Assignment of Other Non-Feedback-Validation Responsibilities 64
Table 15: Assignment of Feedback Validation Responsibilities for Part 1 Architecture Option 1 66
Table 16: Scenarios and Responsibilities Showing Increased Workload or Capability Requirements 69
Table 17: Scenarios Suggesting Stringent Performance and Quality Requirements 71
Table 18: Modified Assignments of Process Model Parts in Part 1 Architecture Option 2 74
Table 19: Modified Assignments of Non-Feedback-Validation Responsibilities for Part 1 Architecture Option 2 75
Table 20: Modified Assignments of Feedback Validation Responsibilities for Part 1 Architecture Option 2 76
Table 21: Examples of Scenarios to be Updated for Architecture Option 2 78
Table 22: UCAs for Updated Situational Information Control Action 79
Table 23: Comparison of Process Model Part Assignments for Part 1 Architectures 82
Table 24: Comparison of SR-1 for Part 1 Architectures 83
Table 25: Comparison of SR-9 and SR-18 for Part 1 Architectures 83
Table 26: Summary of Benefits and Costs for Part 1 Architecture Options 84
Table 27: Sample Assignments of Process Model Parts for Part 2 Architecture Option 1 87
Table 28: Non-Feedback-Validation Responsibilities Assigned to the Pilot for Part 2 Architecture Option 1 88
Table 29: Other Non-Feedback-Validation Responsibilities for Part 2 Architecture Option 1 89
Table 30: Assignment of Feedback Validation Responsibilities for Part 2 Architecture Option 1 90
Table 31: Example Constraints and Scenarios Related to the Use of Feedback 94
Table 32: Responsibilities and scenarios associated with pilot's maintenance of their mental model of the airspace 96
Table 33: Examples of Mental Model Parts Included Due to the Presence of DVE Conditions 98
Table 34: Assignment of Process Model Parts for Part 2 Architecture Option 2 100

Table 35: Assignment of Non-Feedback-Validation Responsibilities for Part 2 Architecture Option 2 101
Table 36: Assignment of Feedback Validation Responsibilities for Part 2 Architecture Option 2 102
Table 37: Sample of Feedback Validation Responsibilities Showing Increased Involvement of ASEC 106
Table 38: SR-16 and its Associated Loss Scenarios 109
Table 39: Assignment of Process Model Parts for Part 2 Architecture Option 3 111
Table 40: Assignment of Non-Feedback-Validation Responsibilities for Part 2 Architecture Option 3 112
Table 41: Example Additional Loss Scenarios Resulting from Unsafe Controller Behavior for UCA-1.2 116
Table 42: Example UCAs for Human Pilot New Control Actions 117
Table 43: Example Scenarios for UCA-1 and UCA-8 118
Table 44: Scenarios Showing Consequences of Inadequate Coordination for Part 2 Architecture Option 3 119
Table 45: Scenarios Showing Consequences of Human Factors Issues for Part 2 Architecture Option 3 120
Table 46: Scenarios Showing Consequences of Unsafe Assumptions in System Design for Part 2 Architecture Option 3 121
Table 47: UCAs for the Piloting Controller 131
Table 48: System-Level Requirements 139
Table 49: Full Details for Decision-Making Strategy, Refined Control Actions and Required Process Model Parts 144
Table 50: Full Details for Feedback Sources 158
Table 51: Full Details for Timing Requirements for Non-Feedback-Validation Responsibilities 162
Table 52: Non-Feedback-Validation Responsibility Assignments for Part 1 Architecture Option 1 166
Table 53: Feedback Validation Responsibilities for Part 1 Architecture Option 1 170
Table 54: Assignment of Non-Feedback-Validation Responsibilities for Part 2 Architecture Option 1 173
Table 55: Assignment of Feedback Validation Responsibilities for Part 2 Architecture Option 1 175
Table 56: Example UCAs for Human Pilot Control Actions 180

Chapter 1 Introduction

1.1 The Role of System Architecture in Systems Engineering

Architecture development is an important step in the system engineering process because the system architecture forms the foundation on which the rest of the system design is based. Figure 1 shows a simplified version of the Systems Engineering Vee Model that illustrates when architecture development is done in the overall system life cycle.

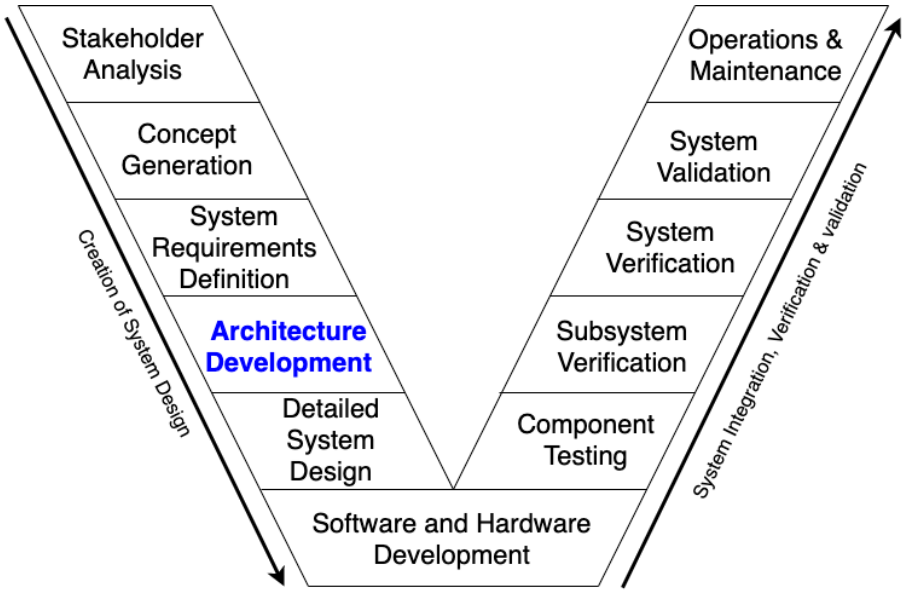


Figure 1: The Systems Engineering Vee model (adapted from [1] - [3])

As shown in Figure 1, after the system requirements are defined, the system architecture is developed before creating the detailed design and developing the software and hardware for the system. The system architecture thus represents the beginning of what will eventually be the complete design for a system and can significantly influence downstream system design activities [4]. Although good system architecture does not guarantee the success of a system, a poor or inadequate system architecture will almost certainly compromise a system's ability to meet its requirements and operate successfully [5].

The system architecture also plays a central role in determining the behavior of a system [4]. Not only does the system architecture define the structure and relationships between the system elements ([2], [4], [6], [7]) but it also defines the fundamental concepts or properties of a system in its environment [6]. As such, architectural decisions can have a lasting impact on properties of the system such as safety. Once the basic architectural decisions have been made, they may be difficult or impossible to change [8]. It is therefore important that the system architecture is created carefully to ensure that the completed system design exhibits all the desired properties and behaviors while minimizing the possibility of undesirable system behavior occurring.

Especially for human-engineered systems, the system architecture also represents a set of design decisions [5] made to solve a design problem and achieve the goals of the system. The architecture development process should therefore support informed decision making by assisting system designers in incrementally obtaining and aggregating information about the system and the behavior and properties it must exhibit [2]. A system designer can then use that information to better understand how the system needs to be designed ([2], [4]), systematically create alternative architectures and evaluate the benefits and tradeoffs between them. This enables more informed decisions to be made when selecting the best architecture for further system development ([2], [9]).

1.2 Limitations of Current Methods for Developing System Architectures

Current methods for architecture development are not adequate for today's increasingly complex sociotechnical systems because they rely on decomposition to create system architectures and manage complexity. This results in three limitations that make developing a good system architecture challenging when they are used to design modern complex systems.

Limited Early Consideration of System Interactions and Emergent Properties

The first limitation is that current methods for developing system architectures incorporate limited consideration of the interactions between system components, especially during the early stages of system design. This is because current methods begin creating system architectures by decomposing systems into their constituent functions and parts before system-level interactions are fully analyzed. As a result, the desirable system interactions that should be achieved or the undesirable system interactions that should be avoided are only analyzed later in the design process. For these same reasons, the properties that emerge from the interactions between system components, which are known as emergent properties, are also not fully analyzed until after an initial architecture has been created. When architectures are created in this way, there may be inadequacies or flaws in the identified system requirements and the system architecture does not exhibit the desired emergent properties such as safety.

Instead of starting with a decomposition of the system into its parts, the architecture development process should begin with an analysis of system interactions and emergent properties as early as possible so that the results can be used to drive architecture development. This ensures that emergent properties such as safety are designed into the system from the beginning and that adequate system requirements are identified to prevent undesirable system behavior from occurring.

Does Not Always Ensure that the Identified Responsibilities Can Be Carried Out Effectively

The second limitation is that current methods for architecture development do not always ensure that the identified responsibilities can be carried out effectively. Current methods for

architecture development create system architectures by decomposing the system requirements into functions or responsibilities and assigning them to the components of the system. The architecture development process therefore begins to focus on the design of individual components before determining how the rest of the system needs to be designed to ensure that the identified responsibilities can be carried out effectively. As a result, supporting requirements may be overlooked and the assigned responsibilities can become more difficult than intended or even impossible to carry out. Consequently, constraints may not be adequately enforced or functions may not be adequately performed and losses may occur.

Instead, the architecture development process should facilitate obtaining more information about what is needed to carry out the identified responsibilities before beginning to create system architectures. This ensures that the architecture development process is informed by not only the responsibilities that the system must perform but also how the system needs to be designed to ensure that the responsibilities can be carried out.

Does not Assist with Managing Complexity and Understanding the System Design

The third limitation of current methods for concept and architecture development is that they do not adequately assist with managing complexity and understanding the system design and its underlying rationale. Today's sociotechnical systems are increasing in complexity as more functionality and performance is demanded of them [5] and they are becoming increasingly interconnected and software-intensive. However, as complexity increases, system designers and reviewers may face difficulty managing that complexity when designing or reviewing a system ([3], [5]). For example, system designers may not be able to consider the entire system or identify all the possible interactions between parts of the system if the number of elements and interactions exceeds their cognitive capabilities. Similarly, those reviewing or assessing the system may not be able to fully understand all the functions of the system or assess whether unsafe interactions have been fully considered if the system complexity is too great.

Current methods for architecture development manage complexity using decomposition of the system and its requirements into hierarchies of parts or functions. As a result, design information is organized in a way that focuses on the breakdown of a system into its constituent elements and contains insufficient information about why that breakdown or system design was chosen. When architectures are created in this way, it becomes more difficult for system designers to understand complex system designs and the underlying rationale or intent information as they develop and refine the system architecture. Similarly, it becomes more difficult for reviewers and assessors to identify flaws or inconsistencies in the system design.

Instead of using decomposition to manage complexity, the architecture development process should use appropriate types of abstraction to guide the design process and help system designers and reviewers to reason about the system design and make informed design decisions. In addition, the design information should be organized in an easily accessible way and include assumptions and design rationale alongside the design information. This ensures that it is as easy

as possible for system designers and reviewers to identify and locate the information they need to understand how the system was designed and why.

1.3 Research Objectives and Approach

This research aims to create a new approach to developing system architectures that addresses the limitations of current methods, can be applied as early as possible and is appropriate for today's increasingly complex systems. This thesis proposes a structured method for creating and assessing architecture options that applies a systems-theoretic approach and appropriate types of abstraction to guide the design process and organize the design information. The proposed method also integrates a hazard analysis method called Systems Theoretic Process Analysis (STPA) into the architecture development process and uses the results to drive the identification of system requirements and inform the creation of a conceptual architecture for a system.

Chapter 2 begins with a summary of current methods for creating system architectures and the reasons that they are no longer adequate for today's complex systems. This is followed by a discussion of the need to instead take a systems-theoretic approach and manage complexity during system design and assessment. Then, a systems-theoretic approach to modeling accident causality called Systems Theoretic Accident Model and Processes (STAMP) and the hazard analysis method called Systems Theoretic Process Analysis (STPA) are described. The application of STPA to creating conceptual architectures is also reviewed. Finally, a human-centered approach to creating system specifications and organizing design information called Intent Specifications is described.

In Chapter 3, the new approach to architecture development is presented. First, an overview of the method is provided and then each step of the method is described along with information about how each step should be carried out.

Finally, in Chapter 4, the new approach is demonstrated using a case study that develops a conceptual architecture for a pilot and automated flight controller performing medevac flights in degraded visual environments (DVEs). Although this research demonstrates this new method for architecture development using an example of medevac flights in DVE, this approach to concept and architecture development can also be applied to other types of complex sociotechnical systems as well.

Chapter 2 Background

2.1 Current Methods for Developing System Architectures

Given the importance of developing a good system architecture, several major organizations that are engaged in systems engineering practice have published guidance documents and standards describing how system architectures should be developed. However, as was outlined in Chapter 1, these methods for developing system architectures are no longer appropriate for today's increasingly complex systems.

This section summarizes the guidance published by the National Aeronautics and Space Administration (NASA), the International Council on Systems Engineering (INCOSE) and the International Standards Organization (ISO) to show how system architectures are expected to be developed today. The shortcomings of these methods will then be discussed to illustrate the need for a new approach to improve our ability to engineer complex systems in the future.

2.1.1 Overview of Current Methods for Developing System Architectures

NASA Systems Engineering Handbook

The NASA Systems Engineering Handbook is intended to provide general systems engineering guidance for anyone working at NASA and describes best practices that should be incorporated into programs and projects of all sizes at NASA [7]. In the handbook, NASA describes the technical processes that are necessary to develop and realize the system using what it calls the Systems Engineering Engine. This Systems Engineering Engine contains three sets of processes: System Design Processes, Technical Management Processes and Product Realization Processes. It is the System Design Processes that are used to create the system design [7].

The System Design Processes begin by defining and baselining a set of stakeholder expectations, the Concept of Operations and the criteria that define how the success of the system will be evaluated. Once the stakeholder expectations have been established, they are decomposed to obtain a set of technical requirements for the project that define all aspects of the system including functional, performance and interface requirements. Once these requirements have been validated, the functions necessary to meet each requirement are identified and then decomposed and allocated to system elements to create a conceptual architecture for the system. It is also at this stage that the functional and subsystem interfaces are defined. The conceptual architecture and technical requirements are then used to perform downstream design activities where the solution is defined and the subsystem specifications are created.

ISO 15288: Systems and Software Engineering – System Life Cycle Processes & INCOSE Systems Engineering Handbook

ISO 15288 is an international standard that establishes a framework for describing the processes that should be carried out in the life cycle of manmade systems. The framework provides generic descriptions of the processes and requirements and is intended to be used by organizations to construct their own life cycle models for their specific product or application [9]. Based on ISO 15288, INCOSE published the INCOSE Systems Engineering Handbook to summarize the process descriptions and requirements contained in ISO 15288 and elaborate on the practices and activities necessary for each process [2]. One part of the system lifecycle processes called the Technical Processes are used to define the system requirements and design the system as well as operate, sustain and eventually dispose of the system. The technical processes begin with the identification of business and stakeholder requirements. These requirements are then transformed into system requirements which are then used to define alternative architecture options. One architecture is then selected to proceed with detailed design and implementation followed by integration, verification and validation before the system is put into operation. After the system is placed into operation, maintenance processes are implemented to sustain the system during operations and disposal processes are carried out when the system is removed from operation at the end of its life ([2], [9]).

The technical processes are used to create the system architecture by starting with identifying the business and stakeholder requirements for a project. Once the business and stakeholder requirements are established, they are used to define system requirements ([2], [9]). This involves starting with the top-level system requirements and decomposing them into lower-level requirements as well as identifying what are called nonfunctional requirements, which are requirements associated with system characteristics such as safety and security [2]. Once the system requirements have been established, they are then used to create alternative architecture options. This is done by first defining the system boundary and the interfaces with external entities before decomposing the system into architectural entities such as functions, system elements or nodes. Concepts, properties or behaviors that are significant for making architectural decisions are also decomposed and allocated to these architectural entities. The architecture is then mapped to the system design by identifying the system elements that correspond to the architectural entities, defining the interfaces and interactions between the system elements and then allocating the requirements to the system elements ([2], [9]). In addition, as part of defining the system architecture, emergent properties should be considered by analyzing the interactions between system elements to identify undesirable interactions that should be avoided and desirable ones that should be reinforced [2].

2.1.2 Shortcomings of Current Methods

The guidance provided in these methods for developing system architectures have several shortcomings that make these methods inadequate for designing increasingly complex, software-intensive systems.

Firstly, system requirements are developed with limited consideration of system interactions and undesirable system behavior. In the methods described in all three documents, the system requirements are developed based on the stakeholder needs and business requirements with minimal guidance on how to ensure that desirable system behavior and properties are achieved and undesirable ones are avoided. It is only after an initial decomposition of the system is performed that these methods begin to fully consider system interactions and undesirable system behavior. Emergent properties are thus also not considered or analyzed until later in the design process. As a result, when these methods begin to fully analyze system interactions or emergent properties, they are either checking for the existence of unintended system interactions or checking if the desired system interactions and emergent properties have been achieved instead of designing them into the system from the beginning.

Modern systems are increasing in both complexity and the number of interactions between components and it is becoming more difficult to use these current methods to identify a complete set of system requirements and avoid undesirable system behavior. This is because modern systems have too many components and interactions to consider if system interactions and undesirable system behavior are analyzed only after an initial decomposition is performed. Furthermore, modern systems are increasingly reliant on interactions between components to achieve the desired system properties. As a result, it is no longer possible to decompose a system into components that are assumed to behave independently and then analyze and design the interactions later. Instead, system interactions and undesirable system behavior should be analyzed as early as possible so that adequate system requirements can be identified at the beginning of the design process. This ensures that desirable system behaviors and emergent properties are designed into the system and undesirable system behaviors are avoided.

Secondly, architectures are created with limited consideration for how to ensure that the necessary responsibilities can be carried out as effectively as possible. In the methods described by all three documents, architectures are created by decomposing the system requirements into sub-requirements. Then, the system is decomposed into its constituent elements and the requirements are allocated to the system elements to create an architecture. By relying on decomposition to create architectures, the architecture development effort becomes focused on designing each component to achieve its assigned responsibilities before more information is obtained about how the system needs to be designed to enable those assigned responsibilities to be carried out as effectively as possible. As a result, supporting requirements that are necessary to carry out a responsibility effectively may be either overlooked or only identified later in the design process. Examples of these include missing feedback or inadequate performance requirements imposed on sensors that the system relies on for feedback data.

Today's systems, however, are expected to exhibit increasingly complex behaviors that require an increasing number of supporting requirements to be satisfied to ensure that those behaviors are achieved. If these supporting requirements are only considered after the system is decomposed, they can quickly exceed the cognitive capabilities of system designers to comprehend and analyze completely. Instead of starting with a decomposition of the system, the architecture should start with an analysis of how the system needs to be designed to achieve the system requirements. That information can then be used to inform architecture development to ensure that the system architecture enables the identified responsibilities to be carried out as effectively as possible.

Finally, these methods for developing system architectures offer minimal guidance on how to manage system complexity and organize design information in a way that assists system designers and reviewers with understanding the system design and the underlying rationale. In all three documents, the methods for architecture development focus primarily on decomposing a system into its parts and functions. Furthermore, while all three documents recognize the importance of documenting assumptions and design rationale, they only offer vague guidance as to how the assumptions and design rationale should be documented. As a result, system architectures are created and documented primarily in terms of their parts and functions with a focus on what the system contains. Information about why the design was chosen is therefore either difficult to find or missing. Consequently, it can be challenging for system designers to locate the design information they need to guide their design decisions or for reviewers to understand the design and its underlying rationale.

When designing highly complex systems, however, the large amount of design information associated with the system makes it even harder to manage complexity and use the design information to understand the system and make informed design decisions. As a result, it becomes even more challenging for system designers to reason about their design when creating and assessing architecture options for a system. Similarly, it also becomes even more challenging for system reviewers or assessors to develop an understanding of the system design and identify any flaws or inconsistencies in it. For these reasons, the architecture development process should instead structure the design process and organize design information using appropriate types of abstraction that facilitate reasoning about the system design. In addition, rationale and assumptions should be documented alongside design decisions. Together, these would assist system designers and reviewers with managing system complexity and make it as easy as possible for them to identify and locate the information they need to understand how the system was designed and why.

2.2 Systems Theory and Emergent Properties

To understand and analyze a system, the approach underlying that analysis must be appropriate for the type of system being analyzed. In his book *An Introduction to General Systems Thinking*, Gerald Weinberg states that there are three types of systems [10] as shown in Figure 2:

1. Those that exhibit organized simplicity
2. Those that exhibit unorganized complexity
3. Those that exhibit organized complexity

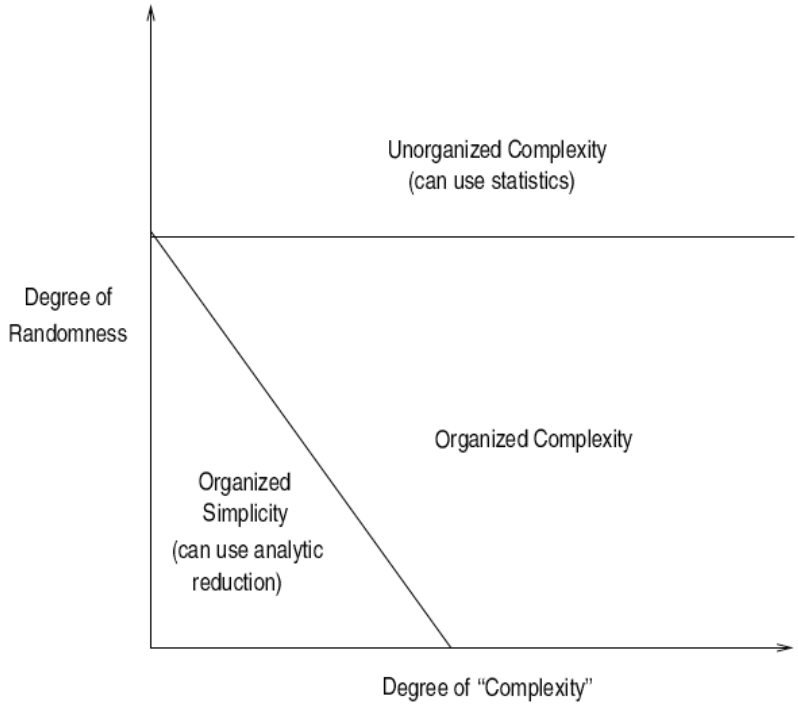


Figure 2: The three types of systems defined by degree of randomness and complexity (from [8])

Some systems involve low complexity and highly structured interactions between the components. These systems are said to exhibit organized simplicity and we have traditionally used decomposition to design and analyze these systems. This approach assumes that each component operates independently and that the behavior of components is the same when examined individually as when they operate as part of an integrated system [8]. However, engineered systems today are too complex for this type of approach to be successful.

Other systems involve high complexity but are uniform and random enough in their behavior that they can be analyzed using statistical or probabilistic approaches. These systems are said to exhibit unorganized complexity. Unfortunately, the new types of complex systems we are building today involve interactions between the components that are not random or uniform and thus statistical or probabilistic approaches are also not appropriate for these systems ([8], [10]).

Instead, modern engineered systems fall into the third category of systems: systems that exhibit organized complexity. Systems that exhibit organized complexity are defined as having both high complexity as well as highly structured interactions between the components. As a result, they are too complex to be analyzed using decomposition and too structured to be analyzed using statistical approaches ([8], [10]).

Systems theory was created to cope with systems that exhibit organized complexity [8]. Instead of analyzing a system statistically or decomposing a system and analyzing the parts separately, a systems-theoretic approach requires consideration of the whole system, including the social and technical aspects. In addition, systems theory states that some properties of a system arise from the structural relationships between the parts of the system as well as the interactions between those parts [8]. Such properties are known as emergent properties and can only be analyzed by considering the entire system and not by only looking at the components. Safety and many other important system qualities in today's systems are emergent. For these reasons, it is necessary to take a systems-theoretic approach and consider the whole system when analyzing modern complex systems to ensure that emergent properties are designed into a system.

2.3 Management of Complexity and Hierarchical Abstraction

As the complexity of modern systems increases, it is becoming more important than ever to manage complexity during both the design process as well when the system is being reviewed. Complexity is not an objective feature of a system and can be managed by viewing the system from a level of abstraction with less resolution [11]. This can be done using the concept of stratified hierarchies to model complex systems in terms of a hierarchy of levels where each level is more complex than the one below. Using this type of model, emergent properties are only characteristic of a particular level and are not reducible to lower levels [12].

There are two main types of hierarchical abstraction: part-whole and means-ends abstraction ([11], [12]). Part-whole abstraction involves breaking the system down into its constituent components or more detailed functions. In part-whole abstractions, the information contained at each level of the hierarchy is refined by information at a lower level. Conversely, information at a lower level is abstracted or aggregated at a higher level. In essence, each level describes *what* the system must do and the lower level describes *how* the system does what it is supposed to do [12]. Functional decomposition or the physical decomposition of a system into its components are commonly used examples of part-whole abstraction.

By contrast, in means-ends abstraction, information contained at each level of the hierarchy describes the goals that the next lower level of the hierarchy must achieve (Figure 3). In other words, the ends described by each level of the hierarchy are achieved using the means described at the next lower level. In this way, each level describes *what* a system must do, the level above describes *why* they must be done and the level below describes *how* they will be done.

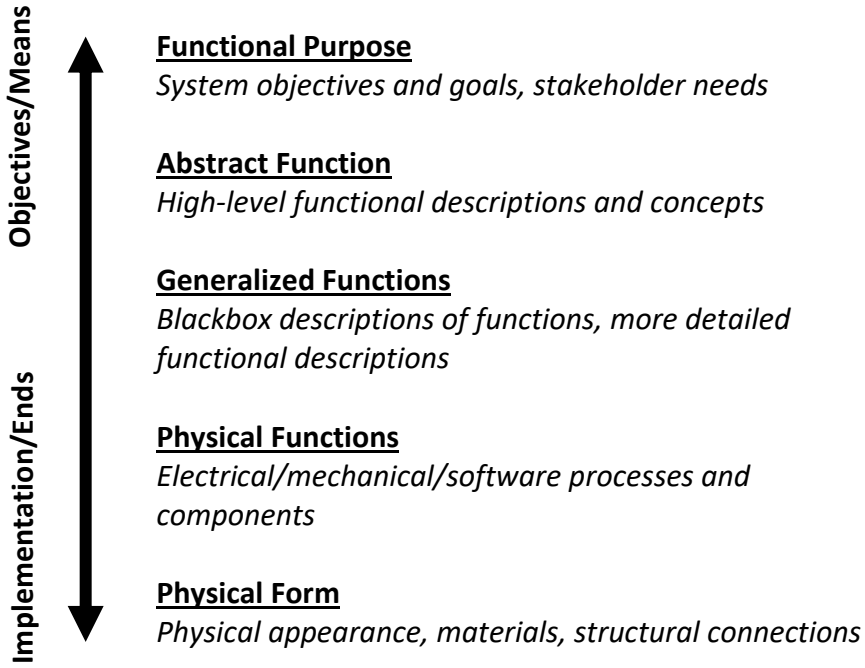


Figure 3: Illustration of means-ends abstraction with examples (adapted from [11])

Instead of representing a system in terms of its parts, means-ends abstraction represents a system in terms of the goals that must be achieved and captures information about intent or design rationale. This intent information is important because it is difficult if not impossible to infer later if this intent information is not identified and recorded during architecture development [12]. Furthermore, the intent information can be used by system designers to guide their lower-level design decisions ([5], [12]) as well as by reviewers to understand and assess the system. Means-ends abstraction can therefore be an especially useful type of abstraction for guiding the design process because it captures intent information and facilitates goal-oriented reasoning about a system [12]. In addition, it can help system designers and reviewers reason about the system and understand the underlying intent as the system is being designed or assessed.

2.4 STAMP and STPA

A system can be analyzed for emergent properties such as safety using a hazard analysis to identify hazards and unsafe behaviors. However, traditional hazard analysis methods and approaches to safety are inadequate for today's complex systems because they were developed 60-75 years ago when systems were simpler than they are today and exhibited at least some of the following characteristics [8]:

- Relatively little use of software
- A relatively traditional relationship between an automated system and the human operators who interact with it
- Accidents largely resulted from component failures

Based on these characteristics, traditional hazard analysis methods (e.g. Fault Trees, Failure Modes and Effects Analysis) use linear chains of events as the underlying accident causality model and focus on the identification and prevention of component failures to ensure the safety of a system. As a result, traditional approaches to safe system designs typically call for the use of redundancy and backup systems to mitigate the effects of component failures [8].

By contrast, current and future systems are increasing in complexity and do not exhibit the same characteristics as older, simpler systems. Modern systems not only increasingly rely on the use of software to carry out their functions but may also have non-traditional relationships between the automated system and the humans who may be interacting with it. For example, the FAA's Urban Air Mobility (UAM) program describes the use of human-automation teaming where human operators must work together in tandem with automation [13] instead of the human operator simply supervising or commanding the automated system. For these reasons, accidents involving these systems have more complex causation than a simple linear chain ([8], [14]). Instead of simply resulting from component failures, accidents may occur due to design flaws, requirements flaws and even organizational factors despite every component performing as intended [14]. For these reasons, traditional safety engineering approaches and hazard analysis methods are no longer sufficient to ensure safety in current and future complex sociotechnical systems [8].

To address these shortcomings and account for these new types of causes of accidents, a relatively new accident causality model called Systems-Theoretic Accident Model and Processes (STAMP) was created. It is based on systems theory and models systems as a whole, including not just the technical aspects such as hardware and software but also humans and other social and organization aspects as well. This holistic view of a system and the interactions between its components enables STAMP to consider other types of accidents and causes that traditional techniques do not consider including non-linear or indirect causes, new kinds of human error, design and requirement flaws and dysfunctional interactions between components in addition to component failures [14]. STAMP also 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 [8]. As a result of this broad view of losses, STAMP-based methods can be used to analyze not only safety and security but also other emergent properties as well [8].

Recognizing that emergent properties such as safety arise from the interactions between the system components, STAMP treats safety as a control problem instead of a component reliability problem [8]. Instead of focusing on maximizing component reliability and availability, accidents or unacceptable losses can be prevented by identifying and enforcing sufficient constraints on a system's behavior and the interactions between the system components as shown in Figure 4.

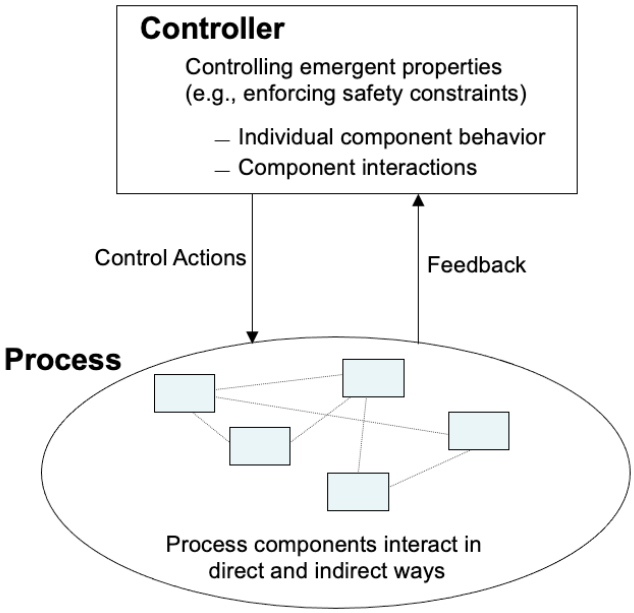


Figure 4: A controller enforcing controls on a system's behavior and the interactions between its components (from [14])

Under this paradigm, a controller enforces these 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 to complete the control loop between the controller and the controlled process. This concept of control is interpreted broadly. Although the controls could be technical or physical controls, they may also be social or organizational controls [14]. The system design must then ensure that these controls and control actions are adequately implemented to enforce the safety constraints for the overall system.

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. Figure 5 shows a generic control structure that includes the control structure during system development (left side) and once the system is operational (right side).

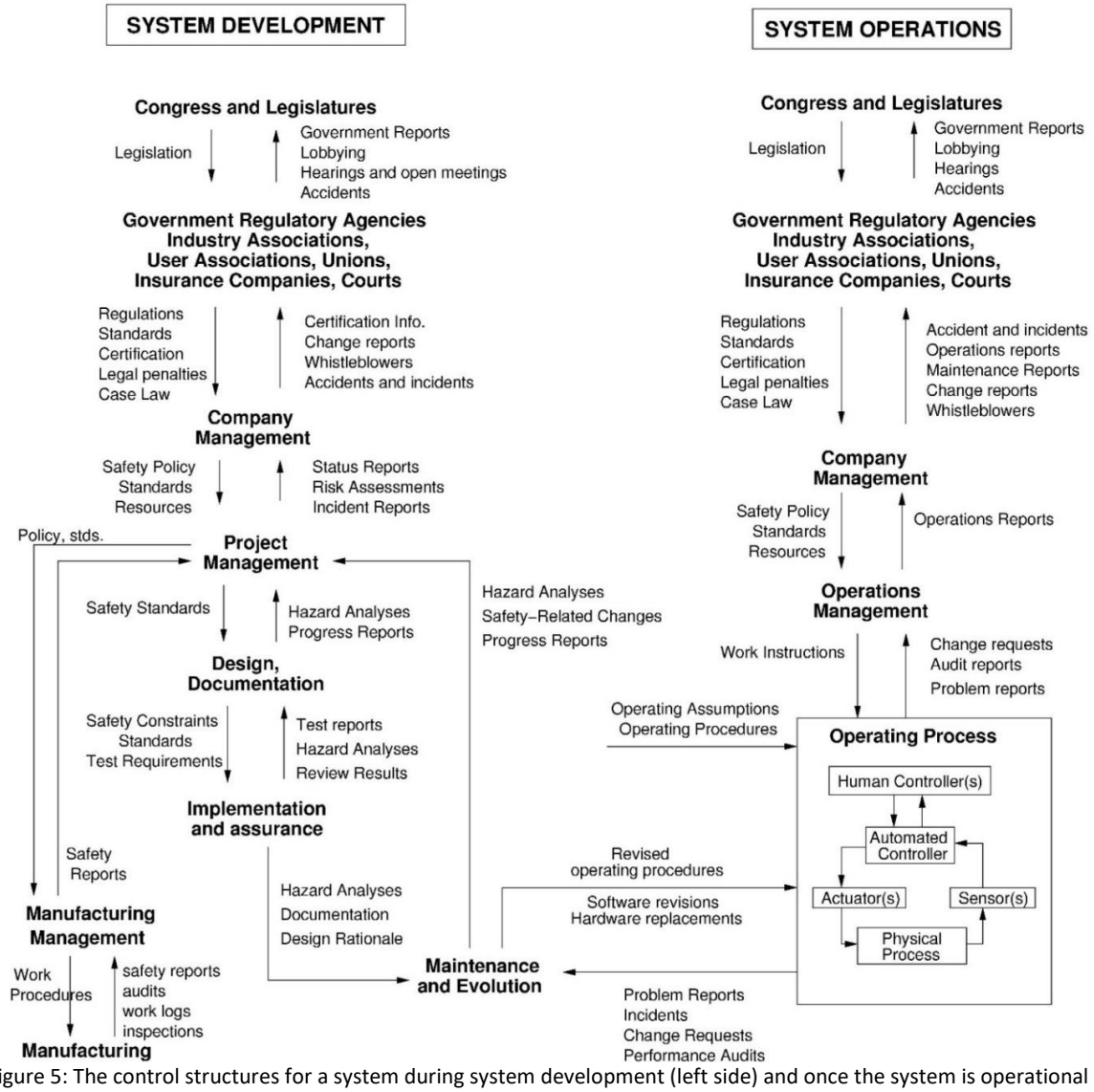


Figure 5: The control structures for a system during system development (left side) and once the system is operational (right side) (from [14])

As shown in Figure 5, the control structure includes not only the human controllers and the system being controlled but also several layers of controllers above them such as project management personnel, government regulatory agencies and personnel involved in operations management and maintenance. This holistic view of the system ensures that the system and the environment in which it is designed, operated and maintained is included in the analysis.

Another important aspect of STAMP is its recognition of the importance of process models and the need for appropriate types of feedback to maintain and update these process models. Every controller needs to have a model of the controlled process that it can use to determine what control actions are necessary to keep the system operating as intended [8] (Figure 6).

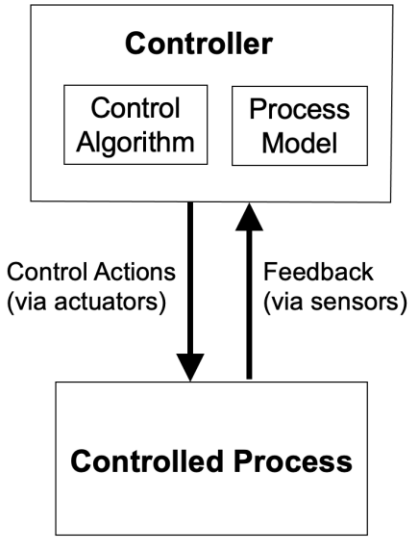


Figure 6: Illustration of a simple control loop involving a controller with a control algorithm and a process model (from [14])

If the controller is automated, this process model may be part of the software running on that controller. For example, the autopilot on a commercial aircraft must have a process model for the current state of the aircraft to decide what flight control inputs to apply to achieve a desired heading and speed. If the controller is human, these process models are usually referred to as mental models that exist in the mind of the human controller. For example, a pilot flying an aircraft requires a mental model of the aircraft and how it works to help them determine what control inputs to provide to the aircraft to achieve the desired flight path [8].

Process models are important for the safe operation of a system because they are used by controllers to make decisions and select appropriate control actions [8]. Controllers must receive adequate types of feedback to keep them updated over time [8]. When there are inconsistencies in the process models, accidents can occur. For example, if a controller's process model of the controlled process is inconsistent with the actual controlled process, the controller may issue a control action that is unsafe in the context of the actual state of the controlled process. Alternatively, if the process models shared between multiple controllers are inconsistent with each other, controllers may issue conflicting control actions or control actions that do not adequately enforce the safety constraints [15]. It is therefore important when designing the system that consideration be given to the process models and the types of feedback that are needed by each controller to maintain an accurate process model of the controlled process and keep it updated. For these reasons, STAMP recognizes that process models are an important part of implementing effective controls to ensure safety.

Based on this theoretical foundation, a new hazard analysis technique called Systems-Theoretic Process Analysis (STPA) was created and Figure 7 shows the basic steps in STPA.

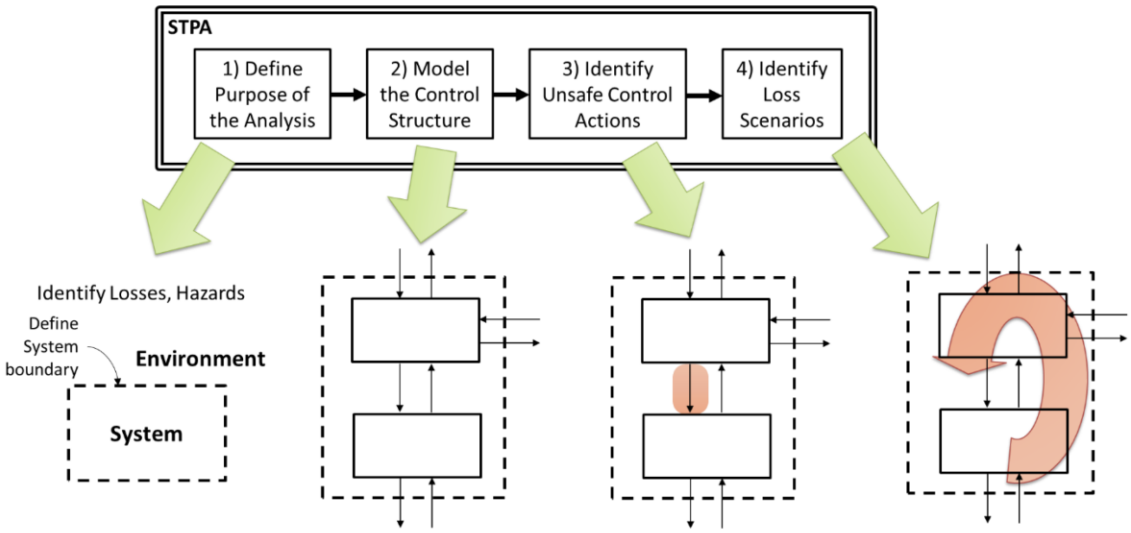


Figure 7: The STPA Process (from [14])

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. As shown in Figure 8, STPA is used to identify Unsafe Control Actions (UCAs) and how they can be caused by flaws in the four main parts of a control loop [14]:

1. Unsafe controller behavior
2. Inadequate feedback or information received by the controller
3. Flaws in the control path
4. Other factors related to the behavior of the controlled process

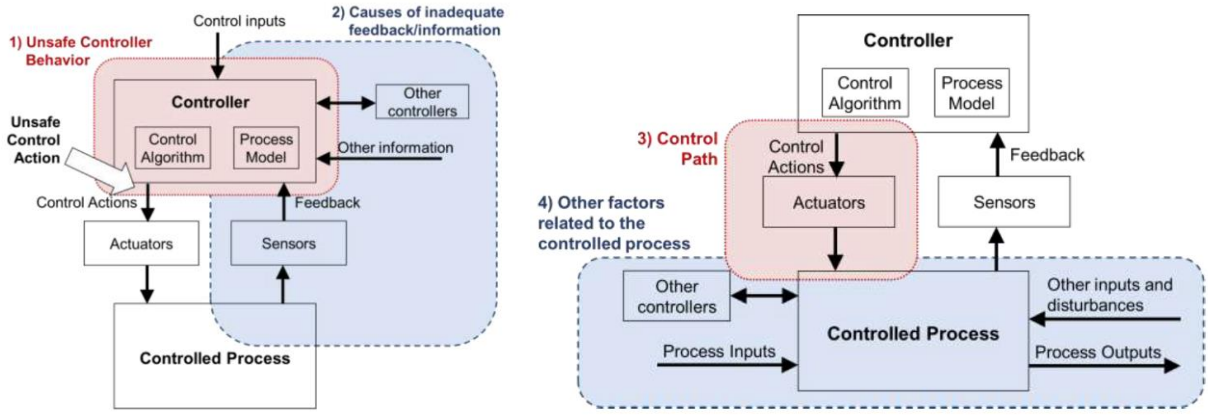


Figure 8: Illustration of how STPA identifies UCAs and the loss scenarios based on flaws in the four main parts of a control loop (from [14])

Since the STAMP model considers new causes of accidents such as requirement errors in addition to component failures, STPA is therefore able to identify many types of hazardous scenarios that are not included or poorly handled by traditional hazard analysis techniques and the analysis results can be used to drive the identification of system requirements that are necessary to prevent the identified scenarios from occurring [8]. In addition to being used to perform analyses

for safety and security, STPA has also been applied to analyze other emergent properties including scalability [16] and maintainability [17].

In addition, because STPA analyzes the system as a whole, including both the technical as well as the human and organizational aspects, STPA is flexible enough to incorporate domain-specific knowledge or knowledge from other engineering disciplines into the analysis of the system. As a result, extensions of STPA have been developed for human controllers supervising automated systems [18] as well as for the analysis of teams of human or automated controllers that must act in a coordinated manner to ensure safe shared control over a system [15]. The extension for human controllers incorporates human factors considerations to refine the model of the human controller in the control structure and it also provides additional guidance during UCA and scenario generation. These can therefore help an analyst to consider possible flaws and errors that might occur when the human operator updates their mental models, in the mental models themselves or during control action selection [18].

2.5 Using STPA to Generate a Conceptual Architecture

In addition to being flexible enough to incorporate knowledge from other engineering disciplines into the analysis, the flexibility of STPA also enables it to analyze systems even during the earliest stages of the design process when little information about the system design is available. When an STPA analysis is incorporated into the design process as early as possible, safety relevant information can be used to drive the design process, allowing safety to be designed into the system from the beginning. This approach was demonstrated by Leveson [3], who showed how STPA can be used to analyze an initial high-level control structure model of the system called a Conceptual Architecture. The STPA analysis results can then be used to inform and guide the refinement of that Conceptual Architecture [3]. Instead of treating STPA as an analysis activity that is separate from the architecture development effort, this method integrates STPA into the architecture development process. The unsafe control actions and loss scenarios identified by STPA can then be used to drive the identification of system requirements and the design of the system. Eventually, the refined conceptual architecture can then be used to create the physical architecture. This method therefore has the potential to minimize the need to change design decisions later by identifying safety-relevant information up-front before making architectural or design decisions.

2.6 Intent Specifications

So far, this chapter has described how systems theory and appropriate types of abstraction as well as STAMP and STPA can be applied to perform top-down, safety-driven architecture development in a way that facilitates informed decision making by system designers and reviewers. However, just developing the concept and architecture using these approaches is not enough and the architecture development process must also be supported by the use of a

specification approach that similarly facilitates gaining an understanding of the system and incorporates intent information and design rationale [3]. To do this, the same concepts of systems theory and hierarchical abstraction can also be applied to the creation of system specifications to ensure that the system can be easily understood and evaluated during both safety assurance and certification activities as well as when maintaining and evolving the system after it is placed into operation

Based on these ideas, Leveson created the concept of an Intent Specification [12]. Intent Specifications are a theoretical approach to system specifications and the information that should be included in them that aims to be human-centered and enhance human understanding and problem-solving. These goals are achieved by organizing information using three types of abstraction: Part-Whole, Refinement and Intent (Figure 9). These types of abstraction are used because they correspond to the different ways in which a user of the specification may want to use the specification to reason about the system and navigate a problem space.

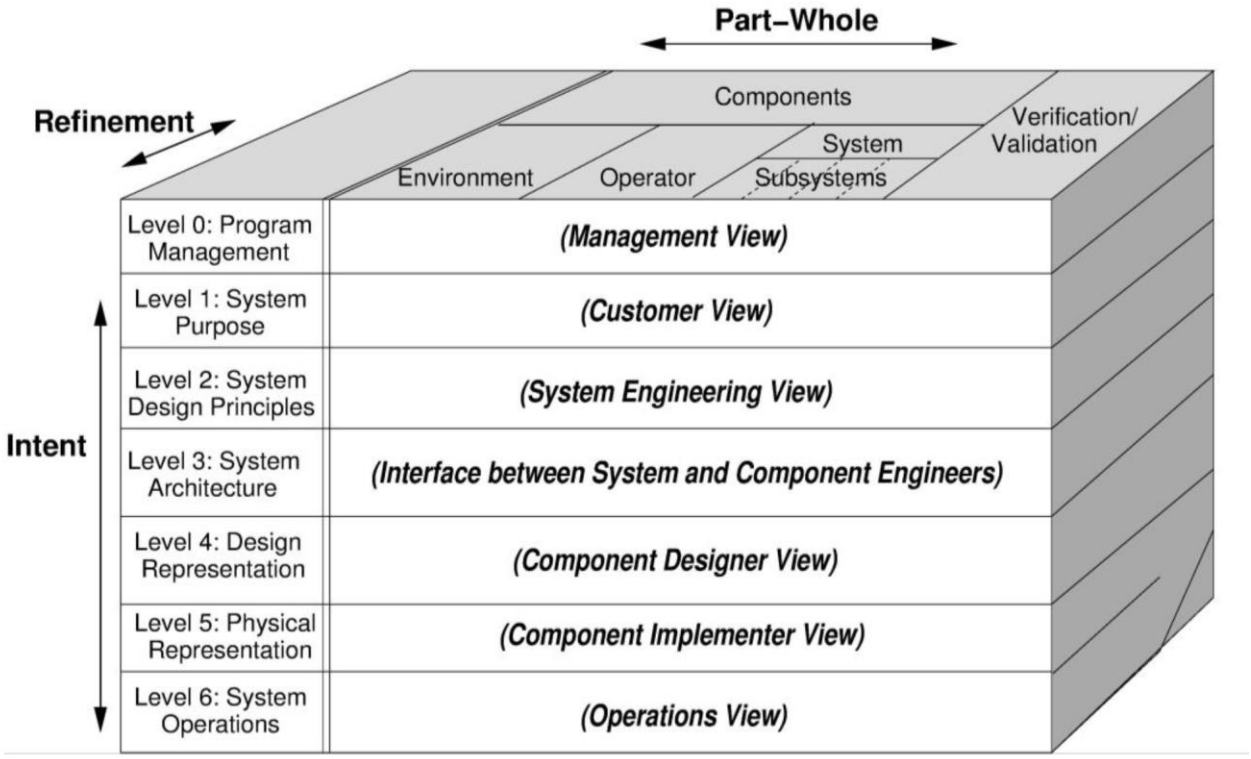


Figure 9: An Illustration of the form of an Intent Specification (from [19])

As shown in Figure 9, the horizontal dimensions represent part-whole abstraction and refinement that allows users to view the system at different levels of detail. The vertical dimension then represents the different levels of intent at which the system can be viewed. The highest levels assist stakeholders and system engineers in reasoning about system goals, requirements and constraints and the lowest levels assist component implementers and system operators in reasoning about how the system works [12]. By organizing information using these three types of abstraction, Intent Specifications can support both top-down and bottom-up approaches to troubleshooting or reasoning about the system, allowing users to easily find the information they

need when they need it to solve a problem, perform system maintenance or change and evolve the system [12].

The vertical dimension of an Intent Specification contains seven levels of intent starting at the Program Management level and ending at the Operations level [12]. At each level, information relevant for that particular perspective of the system is captured and serves as the intent or goals to be achieved by the system at the next lower level. So, for example, information relevant for customers of the system serve as the goals for the system as viewed by the system engineers. This in turn serves as the goals for the system as viewed from the perspective of its architecture and later from the view of the component engineers. In addition, Intent Specifications also emphasize the inclusion of design rationale and assumptions underlying design decisions alongside the design information to assist with understanding why a design decision was made [12]. As a result, Intent Specifications facilitate goal-oriented problem solving by enhancing the user's ability to understand and reason about the system and how it was designed.

Another way in which Intent Specifications enhance human understanding and problem solving is that they combine different types of information relevant to the design of a system into an integrated specification. In addition to the system requirements, safety and other design-related information can all be included in an Intent Specification, allowing it to serve as a central location to store and retrieve all design-related information. This makes design information easier to access when needed instead of distributing that information across multiple documents and repositories [12]. In addition, Intent Specifications contain links between related information contained in other parts of the specification to ensure that related information can be easily located and maintained by the user of the specification when needed [12]. As a result of these features, an Intent Specification enhances human understanding and problem solving because it can serve as the central location to store and retrieve all design-related information and provides traceability to help a user of the specification locate relevant information when needed.

Chapter 3 A New Method for Top-Down, Safety-Driven Architecture Development

3.1 Overview of the Method

The new top-down, safety-driven approach to concept and architecture development being proposed has four steps as shown in Figure 10 that are designed to incrementally guide a system designer in thinking about the system in increasing levels of detail before eventually creating architecture options and assessing tradeoffs.

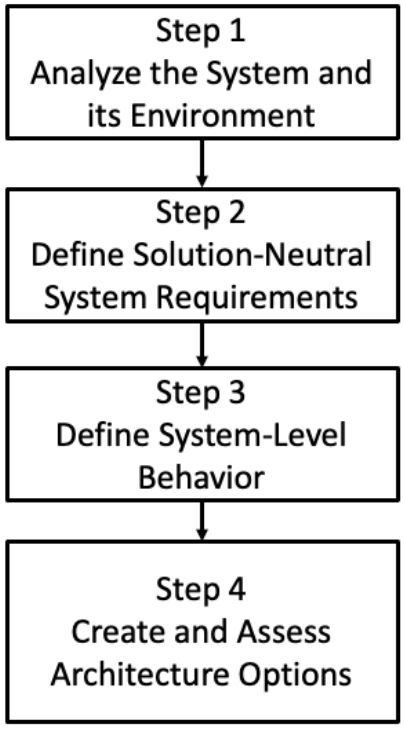


Figure 10: Overview of the new safety-driven approach to concept and architecture development

Although Figure 10 shows the four steps as a linear sequence, this method should not be assumed to proceed in a linear fashion with no allowance for iteration. It is likely that a project implementing this approach will need to iterate on the information generated in each step and revisit earlier steps to make changes as the design process proceeds.

3.2 Step 1: Analyze the System and Its Environment

As shown in Figure 10, the method begins with an analysis of the system and its environment using STPA. It is important that the design process begins with an analysis of the system and its environment before the system itself is designed so that those interactions can be accounted for in the system design. For example, before the architecture of an aircraft is developed, it is important to understand how that aircraft will interact with other aircraft in the airspace as well

as with other controllers such as dispatchers or mission planners. This provides an opportunity to identify any requirements or constraints imposed by the environment on the system or any requirements or constraints that the system might impose on its environment.

3.3 Step 2: Define Solution-Neutral System Requirements

Once the STPA analysis has been performed, the analysis results (i.e. the UCAs and loss scenarios) can be used to define solution-neutral, system-level requirements that will mitigate or eliminate the UCAs and loss scenarios. It is extremely important that these requirements are both solution-neutral and stated at the system level to ensure that the requirements do not make assumptions about the solution or implementation of the constraints at this stage in the architecture development process that later need to be changed. This also avoids unnecessarily constraining the possible solutions or implementations that will fulfill the requirements and successfully enforce the constraints. The requirements should therefore only describe *what* constraints must be enforced to mitigate or prevent the loss scenarios from occurring and should not describe *how* the constraints should be implemented or *who* (i.e. which component in the system) should be enforcing those constraints. As a result, a more systematic and informed decision can be made about which component in the system should fulfill each requirement later in the architecture development process when more information is available.

3.4 Step 3: Define System-Level Behavior

Once the system-level requirements are defined, they become the basis for defining system-level behavior. In this new approach to architecture development, system-level behavior is the behavior of the system that is required to carry out the responsibilities and enable the system to fulfill the system-level requirements that were identified. It is important to note that the system-level behavior is defined without considering the system components that will implement that behavior. This is done to assist system designers with first gaining insight into how the overall system needs to behave before creating and assessing architecture options that will implement that desired system behavior.

The information contained in the system-level behavior describes how a control loop should be designed to ensure that a responsibility can be carried out as effectively as possible to adequately enforce the necessary constraints on the system's behavior. The system-level behavior information therefore consists of five parts that describe the various aspects of how a basic control loop needs to operate. The five parts of system-level behavior are shown in blue in Figure 11 overlayed on top of a basic control loop containing a controller and a controlled process.

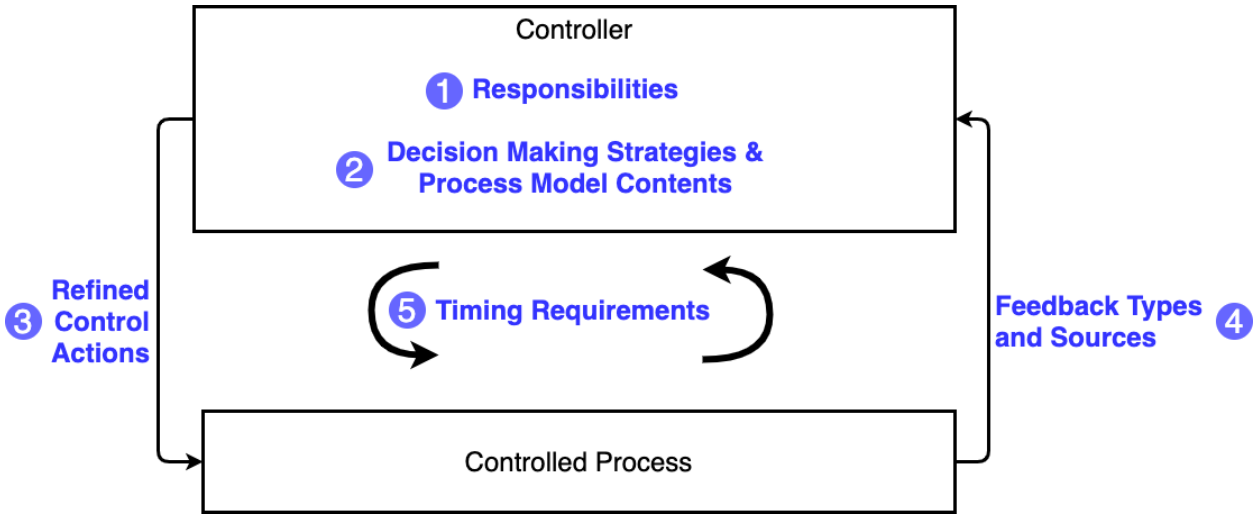


Figure 11: Illustration of the five parts of system-level behavior in blue

Starting at the controller in Figure 11, system-level behavior definition begins by translating the system-level requirements into the responsibilities that the system must carry out. For each responsibility, the decision-making strategy and the process model contents can be defined. The decision-making strategy describes at an abstract level how the system is expected to make the decision(s) necessary to carry out the responsibility. Based on the decision-making strategy, the information that the system needs in its process model to carry out the associated responsibility can be identified. These first two parts of the system-level behavior thus defines how the controller needs to behave to fulfill the system requirements.

The next two parts of the system-level behavior define the control actions and feedback that the controller needs to effectively control the controlled process. Using the decision-making strategy, the control actions can be refined into more specific or detailed ones. The parts of the process model can also be used to identify the feedback necessary to maintain and update the process model parts and the sources of that feedback.

At this point, however, the information identified thus far only offers a static view of this control loop. To include the dynamic and temporal aspects of operating this control loop, the last part of system-level behavior is to identify timing requirements that describe the speed and frequency that each responsibility will need to be carried out by the system.

By defining the aspects of the system-level behavior in this way, each aspect of designing an effective control loop is considered in turn to ensure that the system design implements adequate control over the system's behavior. In addition, because the system-level behavior is driven by the system-level requirements which are in turn derived from the hazard analysis results, this approach to architecture development supports goal-oriented problem solving and ensures that the architecture meets the system-level requirements and implements the necessary constraints to ensure that safety or other emergent properties are achieved. Although defining the system-level behavior information in this order (illustrated in Figure 12) is not strictly necessary, this order can serve as a systematic way to consider each aspect of a control loop and

how it should be designed to enable the responsibilities to be carried out effectively. As before, although Figure 12 is drawn as a linear sequence, iterative refinement of the system-level behavior information is expected.

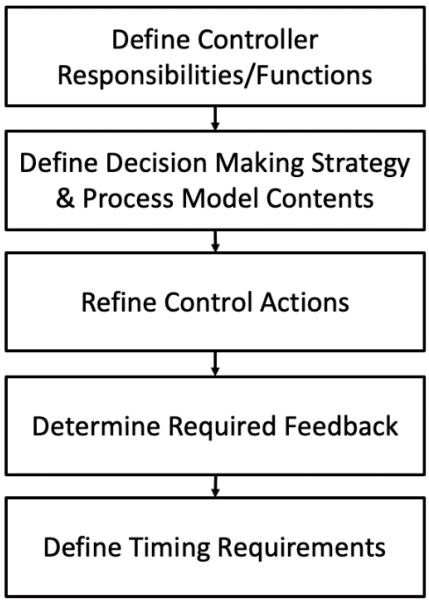


Figure 12: The order in which system-level behavior information should be defined

3.5 Step 4: Create and Assess Architecture Options

Once the system-level behavior information has been generated, it can be used to inform the creation and assessment of architecture options using a 2-part architecture creation process as illustrated in Figure 13.

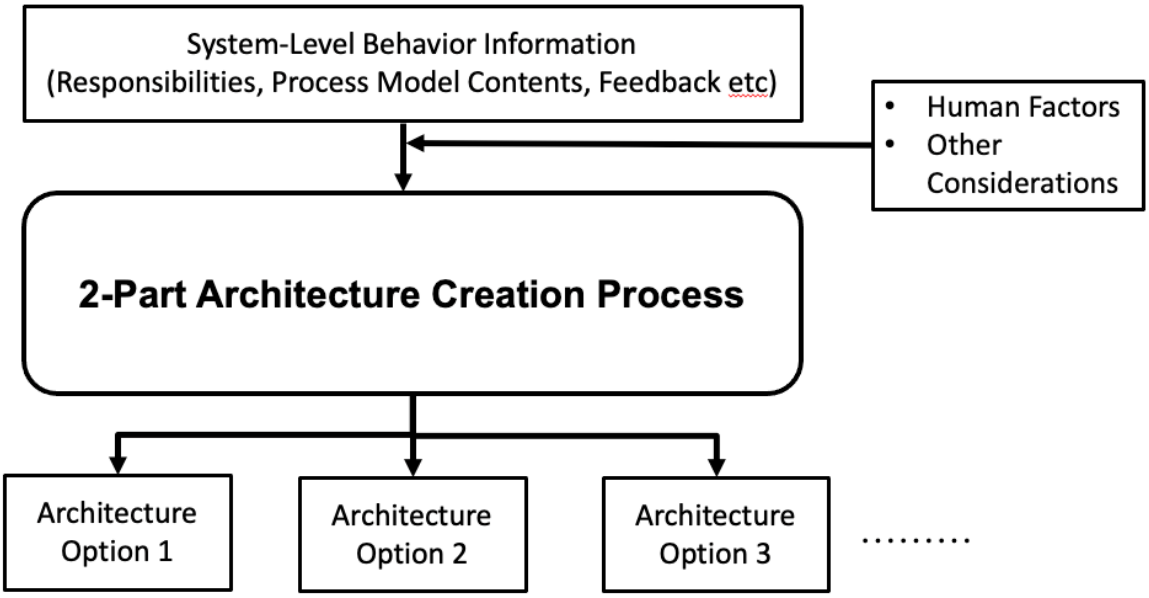


Figure 13: The 2-part architecture creation process and its inputs and outputs

As shown in Figure 13, the system-level behavior information is an input to the architecture creation process. It consists of the set of responsibilities that the system must carry out, the necessary parts of the process model and supporting information about how the system needs to be designed to ensure that the responsibilities are carried out as effectively as possible. This information can now be used in conjunction with human factors and other considerations to inform the creation and assessment of a series of candidate architecture options.

The architecture creation process itself has 2 parts. For each part, a series of candidate architecture options are created by using human factors and other considerations in addition to the system-level behavior information to decide how the process model parts and responsibilities should be assigned to controllers in the architecture. The rationale and assumptions are also recorded alongside each assignment. These architecture options can then be analyzed and the tradeoffs between them assessed to ultimately select one option to carry forward for further system development. This method of creating architecture options ensures that relevant perspectives from other engineering disciplines are incorporated when creating the architecture options and assigning the responsibilities and process model parts.

The 2 parts of the architecture creation process are:

1. Part 1: Create and assess architecture options for the overall system
2. Part 2: Using the selected architecture for the overall system (from part 1), create and assess architecture options for each controller in the overall system

Since the system-level behavior information is defined for the overall system and not for any individual component in that system, the first step in the architecture creation process is to evaluate how those responsibilities and process model parts could be assigned to components in the overall system before proceeding to create architectures for each of the components. To illustrate this, Figure 14 shows a generic example of a system containing three controllers and a controlled process as well as five responsibilities that have been defined for this system.

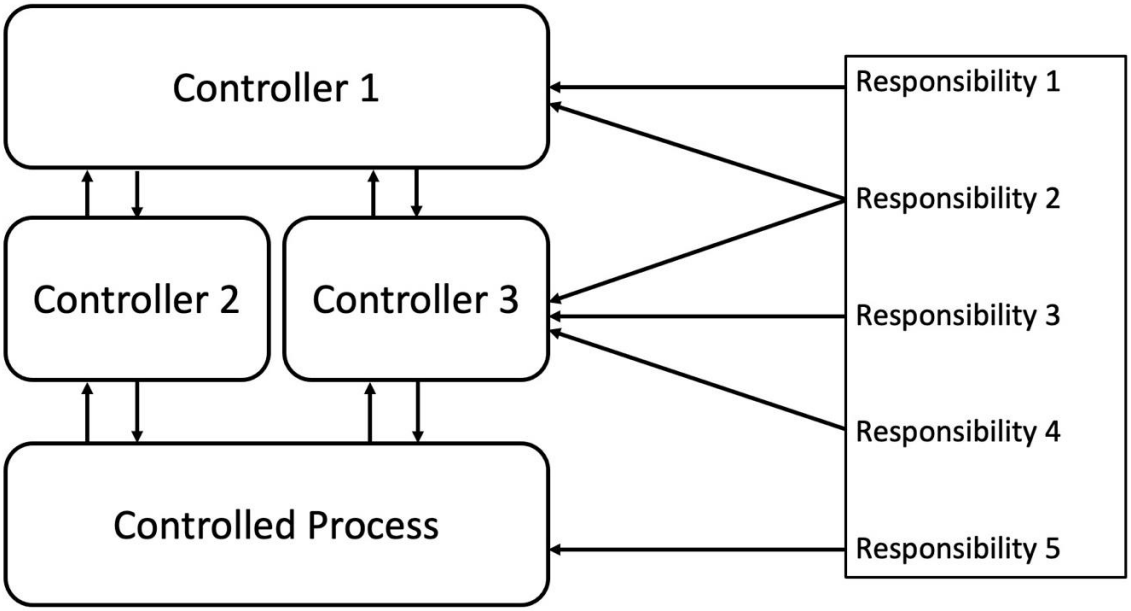


Figure 14: Generic illustration of Part 1 of the Architecture Creation Process

To create an architecture option, the five responsibilities defined for this system must be allocated to the system components. So, as illustrated in Figure 14, one example of an architecture option might be one where:

- Responsibility 1 is assigned to controller 1
- Responsibility 2 is shared between controllers 1 and 3
- Responsibilities 3 and 4 are assigned to controller 3
- Responsibility 5 is assigned to the controlled process

The result of this part of the architecture creation process is the creation of a series of architecture options for the overall system that can then be analyzed individually using further STPA or other analyses to determine the benefits and costs of each option. Based on the tradeoffs that are identified, one of the candidate architecture options will be selected for further system development in the next part of the architecture creation process.

Once the architecture for the overall system has been selected in Part 1, Part 2 creates the architecture options for each component by starting with the process model parts and responsibilities assigned to that system component in the selected architecture from Part 1. Similar to Part 1, these process model parts and responsibilities are then assigned to sub-components within the component that is being designed. Continuing the example from Part 1 that was illustrated in Figure 14, the architecture of controller 3 can be created as illustrated in Figure 15.

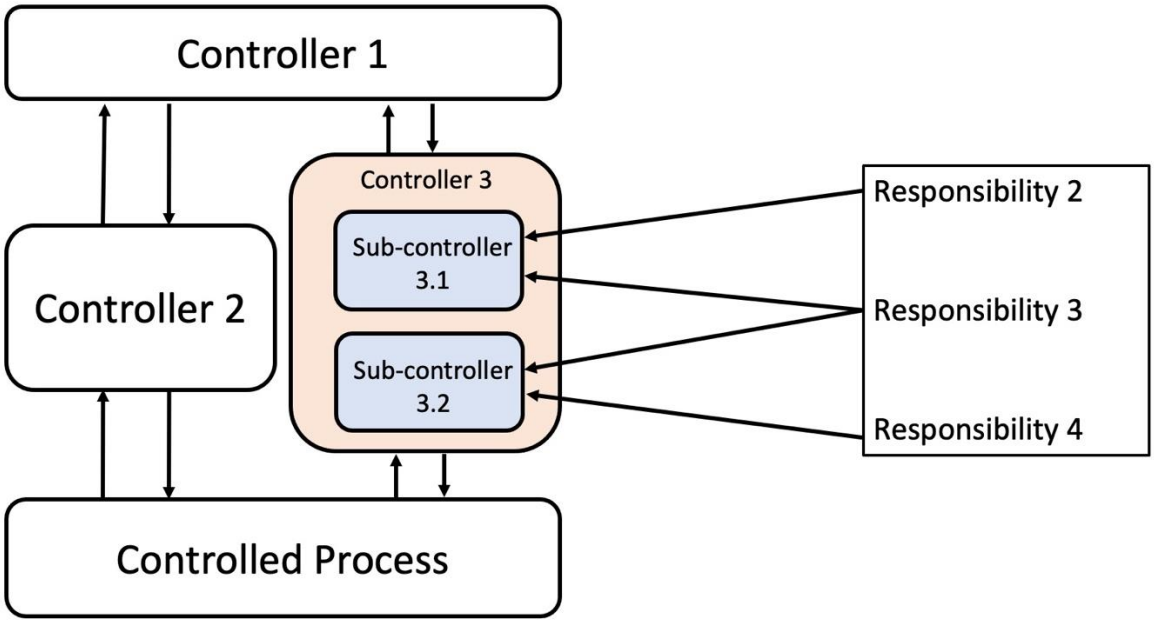


Figure 15: Generic illustration of Part 2 of the Architecture Creation Process

As shown in Figure 15, the architecture for controller 3 starts with responsibilities 2, 3 and 4 that were assigned to it in Part 1 (Figure 14). Architecture options are now created by assigning those responsibilities to sub-controllers 3.1 and 3.2 which are subcomponents of controller 3. One example of a candidate architecture for controller 3 might be:

- Responsibility 2 is assigned to sub-controller 3.1
- Responsibility 3 is shared between sub-controller 3.1 and 3.2
- Responsibility 4 is assigned to sub-controller 3.2

As with Part 1, the result of this part of the architecture creation process is a series of architecture options for the component being designed and these architecture options and the tradeoffs between them can be analyzed using further STPA or other analyses. Based on these analyses, one candidate architecture option will be selected for further development, completing one iteration of refinement of the system architecture.

Since systems can be composed of subsystems that are themselves systems containing further subsystems within them, the definition of a system is recursive [20]. For this reason, this new approach to architecture development can be applied recursively to incrementally refine the system and each subsystem and component contained within it until the detailed system design is complete.

To summarize, this chapter describes a new approach to architecture development and how the concepts and methods reviewed in Chapter 2 were applied. In the next chapter, this new approach to architecture development will be demonstrated for a system that enables medevac flights in poor visibility conditions to be conducted.

Chapter 4 Case Study: Creating a Conceptual Architecture for Performing Medevac Flights in Degraded Visual Environments (DVEs)

4.1 Case Study Overview

In this case study, a conceptual architecture for the human pilot and automated software-enabled controller (ASEC) will be developed for an Emergency Medical Services (EMS) Air Ambulance performing medevac flights in Degraded Visual Environments (DVEs) using the new approach described in Chapter 3.

Although flights under maximum visibility conditions are always preferable, there are emergency situations in which an Air Ambulance service may need to perform a flight to evacuate a patient under reduced or degraded visibility conditions such as rain, fog or snow. Such conditions are known as DVEs [21] and flights under DVE conditions are widely recognized as risky operations for pilots [22] due to the spatial disorientation they can cause [21]. There is therefore a need to find ways to reduce the risks and increase safety for pilots operating medevac flights in DVEs to enable these flights to be carried out as safely as possible.

In response to this need, numerous solutions have been proposed. For example, many aircraft manufacturers are considering the inclusion of heads-up displays and pilot cueing systems [23] as well as synthetic vision systems that integrate the data from aircraft sensors to help pilots re-acquire information they would have otherwise obtained visually and avoid disorientation [24]. Although these solutions are necessary to address specific challenges that pilots face when operating aircraft in DVEs, it is also important to clearly identify the overall system-level requirements that must be satisfied and the responsibilities that the human pilots and the automated systems must perform to ensure safe flight in DVEs.

This case study thus aims to develop a conceptual architecture for this system in a top-down, safety driven manner using this new approach to architecture development, starting with the overall system that is necessary to plan and execute medevac flights safely. This overall system will be called the Flight Operations System (FOS) and is modeled after the procedures and guidelines for air ambulance operations that are published by several public health departments and emergency medical services councils in California ([25], [26]) and New York [27]. Information published by the FAA in its advisory circular regarding air ambulance operations [28] and in title 14 of the Code of Federal Regulations (14 CFR) Part 135 that governs air ambulance operations is also incorporated into this example as needed.

By applying each of the four steps in the new approach, this case study will show how the FOS can be systematically analyzed to identify the system requirements and necessary system-level behavior so that the information obtained can be used to create and assess architecture options for the FOS first and then for the human pilot and ASEC. The case study will then present the

tradeoffs between architecture options that should be considered when selecting a conceptual architecture for the pilot and ASEC to carry forward for further system development.

This case study also organizes the design information that is generated using an Intent Specification. Since this case study represents architecture development occurring during the early stages of system development, the Intent Specification that is created will include information that is primarily contained in levels 1 and 2 of an Intent Specification (i.e. the Customer View and System Engineering View) and will show how information might be structured within each level.

Intent Specification Level 1: Customer View

4.2 Step 1: Analyzing the System and Its Environment

4.2.1 Defining Purpose of the Analysis, System Boundary and System Goal

The purpose of this analysis is to determine how responsibilities can be divided up between the human pilot and Automated Software-Enabled Controller (ASEC) on board an air ambulance to ensure the safe execution of medevac flights in DVEs. The system boundary for this analysis will therefore be called the Flight Operations System (FOS).

The FOS contains the following components:

Table 1: Description of Components in the FOS

#	Component	Description
1	Aircraft Subsystems	This includes all the hardware and components that constitute the physical aircraft that will be flown
2	Piloting Controller	This is an abstracted controller that encompasses both the Human Pilot and ASEC that together must maintain safe and stable flight. Although a conceptual architecture will eventually be created for the human pilot and ASEC as distinct elements of the system, this case study begins by combining them into an abstracted controller so that their architecture can be systematically generated
3	Maintenance Personnel	The personnel that carry out maintenance of the aircraft
4	EMS Operations & Air Traffic Control (ATC)	This group includes local or county EMS dispatch/operations centers, operations centers run by air ambulance service providers as well as standard ATC that controls access to the airspace and manages aircraft throughout their flight

This system has just 1 goal:

G-1: To enable aircraft to be safely flown through an airspace where collision hazards may not be directly visible or visual surveys cannot be conducted due to the presence of man-made or natural obscurants

4.2.2 Initial Assumptions and Constraints

The following is a small set of initial assumptions and constraints that are used in this case study to describe the capabilities that air ambulances are assumed to have and the conditions under which medevac flights might occur.

Data Access Assumptions
These data access assumptions are based on operational procedures described in [28] - [30]:

AA-1. Aircraft may have access to common databases of obstacles, terrain and air traffic in the vicinity of the planned route of flight
AA-2. Aircraft may be able to make and receive updates of obstacle and terrain data in flight
AA-3. Aircraft must have and maintain radio and data communications with local or county EMS operations centers, air ambulance service provider operations centers as well as with ATC

Environmental Assumptions
The following assumptions are made about the environment in which this system will operate as described briefly in [22] and [28]:

EA-1. The system will operate in urban and suburban locations
EA-2. Weather conditions include any of the conditions defined as part of standard Instrument Meteorological Conditions (IMC) that are described in 14 CFR §135
EA-3. The system may operate in both daytime and nighttime conditions

4.2.3 STPA Step 1: Losses, System-Level Hazards & System Constraints

The first step in performing an STPA analysis is to list the losses, system-level hazards and system constraints for the FOS.

Losses

The losses that are relevant for the FOS are:

L-1: Loss of life or injury
L-2: Loss or damage to aircraft or equipment
L-3: Nonachievement of mission

Hazards

From these losses, the following hazards are derived:

H-1: Aircraft is uncontrollable [L-1, L-2, L-3]
H-2: Structural integrity of the aircraft is violated [L-1, L-2, L-3]
H-3: Violation of minimum aircraft separation standards [L-1, L-2, L-3]
H-4: Operating the aircraft is harmful to human health [L-1]
H-5: Aircraft is unable to conduct mission tasks [L-3]

System Constraints

The following are the system constraints that are derived from the system-level hazards:

SC-1. The aircraft must remain controllable by the piloting controller at all times [H-1]
SC-2. Aircraft airframe integrity must be maintained at all times [H-2]
SC-3. Aircraft must satisfy minimum separation requirements from other aircraft and objects [H-3]
SC-4. If the aircraft violates minimum separation requirements, the violation must be detected and measures taken to prevent collision [H-3]
SC-5. Aircraft environment must remain safe for human occupants of the aircraft at all times [H-4]
SC-6. Aircraft must be able to conduct mission tasks [H-5]

4.2.4 STPA Step 2: Safety Control Structure

Once the losses, system-level hazards and system-level constraints have been identified, the next step is to model the system using a safety control structure. Figure 16 shows the control structure for the FOS where each of the four components within the system boundary as listed in Section 4.2.1 are shown as controllers in the control structure along with the control actions and feedback that are passed between them. In addition, since a key aspect of operating aircraft in DVEs is the ability to quantify weather conditions and detect other objects and aircraft in the airspace, the operational environment is included in this control structure and provides direct feedback to both the pilot who makes direct visual observations and to the aircraft subsystems where sensors are used to sense the environment.

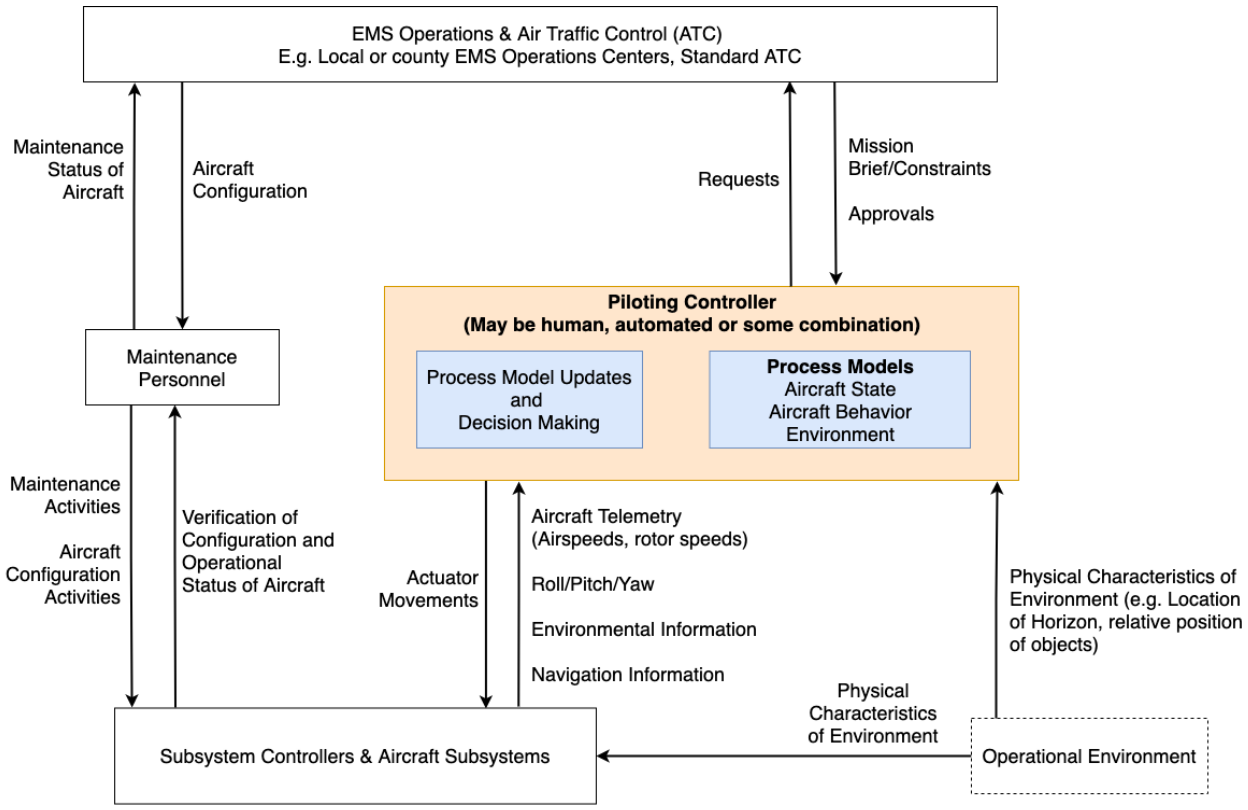


Figure 16: Control Structure with Abstracted Piloting Controller Highlighted

Since this STPA analysis will be used to generate system-level requirements for the FOS and create a conceptual architecture for the FOS and Piloting Controller, this control structure starts at a high level of abstraction so that the system and its environment can be first considered as a whole. For these same reasons, the human pilot and automated controllers on-board the aircraft are combined into an abstracted controller called the Piloting Controller. By doing this, the initial STPA analysis can be used to identify all the responsibilities and control actions needed for this abstract controller before dividing the responsibilities up between a human and automated controller. The benefits and tradeoffs of the different ways to divide up these responsibilities can then be analyzed later when more information is available.

Initial Safety Responsibilities

Even during the early stages of system development, it is possible to define an initial set of responsibilities, albeit at a high level of abstraction, that must be enforced by the components in the control structure to ensure safety. As the STPA analysis proceeds, these responsibilities will be gradually refined. The initial list of high-level responsibilities for each component of the FOS includes the following:

R-1. Aircraft Subsystems

- R-1.1. Execute actuator movements commanded by the Piloting Controller to control the aircraft [SC-1]
- R-1.2. Provide information regarding the telemetry of the aircraft (e.g. airspeeds, physical position) to the Piloting Controller [SC-1]
- R-1.3. Provide information regarding the position and speed (when relevant) of other aircraft and objects in the vicinity of the aircraft to the Piloting Controller [SC-3, SC-4, SC-6]

R-2. Maintenance Personnel

- R-2.1. Ensure that the aircraft receives required preventative maintenance [SC-1]
- R-2.2. Ensure that all maintenance is completed and personnel clear of the aircraft when the aircraft is ready to depart [SC-3, SC-5, SC-6]
- R-2.3. Ensure that the aircraft is configured appropriately [SC-1, SC-6]

R-3. Piloting Controller

- R-3.1. Control the aircraft to satisfy minimum separation requirements from other aircraft and objects at all times [SC-3]
- R-3.2. Maintain an updated model of the environment, state of the aircraft and the state of the airspace around the aircraft throughout all phases of flight [SC-1, SC-3]
- R-3.3. Ensure that actuator commands sent to the aircraft subsystems does not exceed the capabilities of the airframe [SC-2]
- R-3.4. Manage and control the aircraft to carry out the mission [SC-6]

R-4. EMS Operations and ATC

- R-4.1. Provide the Piloting Controller with all available information needed to safely control the aircraft [SC-1, SC-2, SC-3, SC-6]
- R-4.2. Coordinate air traffic in controlled airspaces and maintain a coherent operating picture of controlled airspaces [SC-1, SC-3, SC-4, SC-5, SC-6]
- R-4.3. Provide the Piloting Controller with all mission information and constraints to successfully execute the mission [SC-6]

4.2.5 STPA Step 3: Unsafe Control Actions (UCAs)

The third step in STPA is to analyze the control actions in the control structure to identify unsafe control actions (UCAs) and the associated controller constraints necessary to prevent the UCAs from occurring. Table 2 shows some examples of UCAs that were identified for the Actuator Movements control action provided from the Piloting Controller to the Aircraft Subsystems. Additional UCAs for the Actuator Movements control action can be found in Appendix A.

Table 2: Example UCAs for Actuator Movements Control Action

Control Action	Providing	Not Providing	Too Early/Too Late	Applied Too Long/Stopped Too Soon
Actuator Movements	**UCA-1.1:** Piloting Controller provides actuator movements during takeoff when the aircraft is not in a safe departure state [H-3, H-4, H-5] **UCA-1.2:** Piloting Controller provides actuator movements that steers the aircraft toward another aircraft or object [H-3]	**UCA-1.5:** Piloting Controller does not provide actuator movements when violation of minimum separation is imminent [H-3] **UCA-1.6:** Piloting Controller does not provide actuator movements during critical phases of flight [H-1, H-3]	**UCA-1.7:** Piloting Controller provides actuator movements too late during takeoff after takeoff clearance is granted when another aircraft or obstacle has entered the airspace near the aircraft [H-3] **UCA-1.8:** Piloting Controller provides actuator movements too late when the aircraft is near another aircraft or obstacle [H-2, H-3]	**UCA-1.10:** Piloting Controller applies actuator movements for too long during takeoff after it has passed the desired altitude [H-1, H-3] **UCA-1.11:** Piloting Controller stops providing actuator movements too soon during takeoff before the desired altitude is reached [H-3]

4.2.6 STPA Step 4: Identify Loss Scenarios

The next step in STPA is to identify the loss scenarios based on these UCAs. To do this, a new approach to scenario generation described by Cabosky [29] was used to help manage the potentially large number of scenarios that would be generated and reduce repetition. In addition, this new approach to scenario generation should help to ensure that as many causal factors as possible are considered in the loss scenarios. This is important because the system-level requirements will be generated in later steps based on these loss scenarios and therefore good coverage of the causal factors by the loss scenarios should ensure that the set of system-level requirements obtained will be as complete as possible. This new approach to scenario generation uses guide words based on four basic scenario types to assist with identifying loss scenarios [29]:

1. Unsafe Controller Behavior
2. Unsafe Feedback Path
3. Unsafe Control Path
4. Unsafe Controlled Process Behavior

Using this new approach to scenario generation, a wide variety of different scenarios were identified and some examples of loss scenarios are identified for UCA-1.2 (Table 3) and UCA-1.5 (Table 4). More examples of loss scenarios are shown in Appendix B.

UCA-1.2: Piloting Controller provides actuator movements that steers the aircraft toward another aircraft or object [H-3]

Table 3: Example Loss Scenarios for UCA-1.2

Scenario Type	Scenario ID	Scenario
Unsafe Controller Behavior	CS-1.2.1-1.1	Despite receiving feedback that there was another aircraft or object nearby, DVE conditions cause more noise in sensor data than is normally present, making the useful feedback more difficult for the piloting controller to distinguish from the noise or increasing the likelihood that the piloting controller wrongly decides that the sensor data shows no useful feedback/detections. As a result, the piloting controller has the wrong process model of the state of the environment, the aircraft or the airspace around the aircraft and wrongly believe that the airspace nearby the aircraft is clear to pilot toward. The piloting controller therefore selects actuator movements that pilot the vehicle toward the other aircraft or object [SLR-13, SLR-19]

Scenario Type	Scenario ID	Scenario
Unsafe Controller Behavior (Cont'd)	CS-1.2.1-1.4	Despite receiving feedback that there was another aircraft or object nearby, the piloting controller receives an input from another aircraft or external controller that indicates that the airspace nearby the aircraft is clear. If the pilot assumes that the external controller/input is providing accurate information, they may disregard the seemingly conflicting feedback that does indicate that another aircraft or object is nearby. As a result, the piloting controller wrongly updates its process model of the airspace nearby the aircraft and selects actuator movements that pilot the vehicle toward the other aircraft or object [SLR-3, SLR-15]
	CS-1.2.1-3	Despite receiving feedback that there was another aircraft or object nearby, the piloting controller is forced to make a quick decision to avoid violating minimum separation. In making this quick decision, the piloting controller does not fully account for all objects and aircraft in the airspace. As a result, the piloting controller tries to avoid one object/aircraft and collides with another aircraft/object instead. [SLR-17, SLR-18]
Unsafe Feedback Path	CS-1.2.2-1	The piloting controller receives feedback that did not indicate that another aircraft or object was nearby because DVEs degrade or obscure sensors or the sensor suite was not designed to operate in the current DVEs. As a result, the piloting controller receives wrong, incomplete or missing feedback about the environment, DVE conditions or the state of the aircraft which the piloting controller uses to update its process model, leading to wrong process models [SLR-2, SLR-15, SLR-19]
Unsafe Control Path	CS-1.2.3-1	The piloting controller does not provide actuator movements that pilots the aircraft toward another aircraft or object but actuator movements to do so are received by the aircraft because an adversary spoofs the actuator movements sent from the piloting controller and causes the aircraft to believe the piloting controller has sent actuator movements [SLR-11]
Unsafe Controlled Process Behavior	CS-1.2.4-1	Actuator movements are not received by the aircraft but the aircraft still violates minimum separation because DVE conditions (e.g. wind gusts) move the aircraft toward the other aircraft or object with enough force or at a high enough rate that the piloting controller is unable to react quickly enough or with appropriate amplitude to correct the disturbance [SLR-22]

UCA-1.5: Piloting Controller does not provide actuator movements when violation of minimum separation is imminent [H-3]

Table 4: Example Loss Scenarios for UCA-1.5

Scenario Type	Scenario ID	Scenario
Unsafe Controller Behavior	CS-1.5.1-1.1	Despite receiving feedback that violation of minimum separation is imminent, the piloting controller is unable to select new actuator movements quickly enough to avoid violation of minimum separation. This might occur if the piloting controller recognizes the imminent violation too late or takes too long to select new actuator movements. As a result, the piloting controller is unable to select new actuator movements before the violation of minimum separation occurs [SLR-23]
Unsafe Feedback Path	CS-1.5.4-1	Actuator movements are received by the aircraft but the aircraft still violates minimum separation because the piloting controller may have the wrong process model of the environment conditions and selects actuator movements that are insufficient to effect the desired change in flight path [SLR-25]

The example scenarios shown in Table 3 and Table 4 thus illustrates the broad range of causal factors that are considered in the loss scenarios. Not only do the loss scenarios consider how unsafe behavior at different points in the control structure can lead to a hazard or loss but they also consider how the environment the system operates in (e.g. DVE conditions, external inputs) could contribute to the occurrence of a hazard or loss. For example, CS-1.2.1-1.1 and CS-1.2.2-1 describe how the effects of DVEs on feedback could lead to inadequate or conflicting feedback. CS-1.2.1-3 and CS-1.5.1-1.1, on the other hand, describe how unsafe decision-making could lead to an unsafe control action being issued. Even cybersecurity considerations can be included in scenarios such as CS-1.2.3-1. In addition, multiple causal factors can contribute to the occurrence of a loss or hazard. For example, in CS-1.2.4-1 and CS-1.5.4-1, DVE effects combined with inadequate decision-making can collectively lead to an unsafe control action. These loss scenarios therefore demonstrate the wide variety of causal factors that can and should be considered prior to identifying the necessary system-level requirements.

Once the loss scenarios have been generated for all UCAs, the first step in this new approach to architecture development is complete. In the subsequent steps, these STPA results will be used to define requirements and system-level behavior information that will inform the creation and assessment of architecture options.

4.3 Step 2: Defining Solution-Neutral, System-Level Requirements

The next step in this new approach to concept and architecture development is to identify the solution-neutral, system-level requirements that are necessary to ensure safety and prevent the UCAs and loss scenarios identified in the STPA analysis conducted in Step 1 from occurring. As discussed in Chapter 3, it is important that these requirements are both solution-neutral and stated at the system level to ensure that these requirements describe *what* constraint must be enforced instead of *how* that constraint should be implemented or *who* should be enforcing that constraint.

The solution-neutral, system-level requirements are generated by identifying one or more suitable constraints that, when enforced by some means, would mitigate or prevent the scenarios from occurring. The identified constraints are then written in standard requirements language (e.g. "shall" statements). Figure 17 and Figure 18 demonstrate how these solution-neutral, system-level requirements are generated using two examples of a UCA and loss scenario and their associated requirement.

UCA-1.2: Piloting Controller provides actuator movements that steers the aircraft toward another aircraft or object [H-3] **CS-1.2.2-1:** The piloting controller receives feedback that did not indicate that another aircraft or object was nearby because DVEs degrade or obscure sensors or the sensor suite was not designed to operate in the current DVEs. As a result, the piloting controller receives wrong, incomplete or missing feedback about the environment, DVE conditions or the state of the aircraft which the piloting controller uses to update its process model, leading to wrong process models [SLR-2, SLR-15, SLR-19]
Requirement: The FOS must receive all data required to determine the state of the environment and conditions around the aircraft under all DVE conditions at all times *Rationale/Assumptions: This requirement ensures that the piloting controller has the information it needs to make decisions and is never making decisions by using guesses or assumptions that might turn out to be incorrect. This requirement assumes that it is possible to make this guarantee without exception using a combination of engineering design techniques when creating the detailed system design. If this assumption is invalidated, the associated causal scenario may not be fully prevented and further requirements may be necessary to avoid unsafe system behavior.*

Figure 17: Example 1 of how solution-neutral, system-level requirements are generated

UCA-1.2: Piloting Controller provides actuator movements that steers the aircraft toward another aircraft or object [H-3]

CS-1.2.4-1: Actuator movements are not received by the aircraft but the aircraft still violates minimum separation with the nearby aircraft or object because DVE conditions (e.g. wind gusts) move the aircraft toward the other aircraft or object with enough force or at a high enough rate that the piloting controller is unable to react quickly enough or with appropriate amplitude to correct the disturbance [SLR-22]

Requirement: The FOS must be able to respond quickly enough and with an appropriate magnitude to disturbances to prevent unintended movement of the aircraft.

Rationale/Assumptions: This requirement ensures that clear expectations are established as to how quickly the FOS must be able to respond to disturbances and how much disturbance the FOS must be able to reject to maintain safe flight. This requirement assumes that it is possible to define a uniformly-applicable set of worst-case conditions under which the FOS must be able to maintain safe and stable flight. If this is not possible, this requirement may have to be modified to account for multiple types of worst-case DVE conditions.

Figure 18: Example 2 of how solution-neutral, system-level requirements are generated

As shown in Figure 17 and Figure 18, the two identified requirements are stated at the system level because they simply state what the FOS as a whole must do instead of identifying a particular component of the FOS. The two identified requirements are also solution-neutral because they simply state what constraint is necessary to mitigate or prevent the loss scenario and do not state how the constraint will be implemented or achieved.

It is also worth noting that, since the definition of these requirements represents one type of design decision that is being made, it is important to record the rationale and assumptions so that anyone reviewing or modifying the system in the future can understand why the requirement was generated and evaluate the validity of the underlying assumptions.

By applying this method for each causal scenario in the STPA analysis results, a full set of system-level requirements can be generated. Table 5 shows the set of requirements generated for this case study as a simple list for compactness and the full details for these requirements including the underlying rationale and assumptions can be found in Appendix C.

Table 5: System-Level Requirements List

Req ID	System-Level Requirement
SLR-1	The FOS must be able to determine if any feedback it is receiving about the aircraft's mission readiness, state of the aircraft and the airspace around it is too old
SLR-2	The FOS must account for prevailing DVE conditions and their possible effect on feedback sources when making use of feedback information
SLR-3	The FOS must verify the accuracy of any inputs from another aircraft/controller before updating process models based on that input
SLR-4	The FOS must confirm that all aspects of the aircraft state match the safe departure state prior to departure
SLR-5	The FOS must always be able to determine if the state of the aircraft changes between the time that it was checked and departure
SLR-6	The FOS must confirm that data provided to it about the expected safe departure state of the aircraft has been fully specified
SLR-7	The FOS must ensure that all relevant personnel are aware of any changes to the expected safe departure state of the aircraft
SLR-8	The FOS must process all feedback to make a deliberate decision if it is to be ignored/dropped
SLR-9	The FOS must ensure that it is receiving all data required to determine the expected safe departure state and determine the current state of the aircraft under all DVE conditions at all times
SLR-10	The FOS must ensure that flight crews are aware of the most updated state of any maintenance tasks on the aircraft
SLR-11	The FOS must ensure that only authorized actuator movements are executed
SLR-12	All FOS equipment and systems must be able to operate in the expected DVE conditions
SLR-13	The FOS must be able to distinguish useful detections/feedback from the noise that might be present in feedback data under all DVE conditions at all times
SLR-14	The FOS must be able to process sensor data and use it to update its process model sufficiently quickly (within TBD seconds)
SLR-15	The FOS must receive all data required to determine the state of the environment and conditions around the aircraft under all DVE conditions at all times
SLR-16	The FOS must always be able to determine if the state of the airspace changes between the time that it was checked and the commencement of a maneuver
SLR-17	The FOS must take into account the current and future movements of other aircraft in the vicinity when selecting actuator movements
SLR-18	The FOS must ensure that aircraft movements are selected such that sufficient reaction time is available to react and prevent violation of minimum separation if intent or movements of the other aircraft are different than expected
SLR-19	The FOS must be able to detect all objects and other aircraft in the environment under all DVE conditions at all times

Req ID	System-Level Requirement
SLR-20	The FOS must be able to detect slow or subtle changes in the state of the aircraft under all DVE conditions at all times
SLR-21	The FOS must select actuator movements that minimize the risk of violation of minimum separation when information about the state of the airspace is in a degraded condition (e.g. delayed)
SLR-22	The FOS must be able to respond quickly enough and with an appropriate magnitude to disturbances to prevent unintended movements of the aircraft
SLR-23	The FOS must be able to respond quickly enough to avoid violations of minimum separation
SLR-24	The FOS must ensure that a viable path is always available to avoid a violation of minimum separation
SLR-25	The FOS must select actuator movements that are sufficient and appropriate to effect the desired change in flight path under the given environmental conditions
SLR-26	The FOS must ensure that actuator movements avoid all possible violations of minimum separation

4.4 Step 3: Defining System-Level Behavior

The next step in this new approach to concept and architecture development is to use the system-level requirements to define the system-level behavior information that describes what is necessary from a control loop perspective (Figure 11) to ensure that the constraints described by the requirements are adequately enforced by the system. Table 6 summarizes how each aspect of the system-level behavior information should be defined.

Table 6: Descriptions of the Five Aspects of System-Level Behavior Information

System-Level Behavior Aspect	Description
Define Controller Responsibilities	Translate the system-level requirements into responsibilities
Define decision-making strategy	Describe at a high level of abstraction how a controller makes a decision
Refine Control Actions	Refine abstracted control actions into more specific ones
Determine process model parts and required feedback	Identify the process model parts needed to support decision-making and the sources of feedback needed to support process model updates
Define timing requirements	Determine how often and how quickly a function must be carried out

For each system-level requirement, these five aspects must be defined to obtain the system-level behavior for the system being designed. The following sub-sections will show how to generate the system-level behavior information using requirements SLR-19 and SLR-23 as examples. Full details for the system-level behavior generated for this case study can be found in Appendix D.

4.4.1 Defining Responsibilities

The first step in defining system-level behavior information is to translate each requirement into a responsibility that the system must carry out. Like the system-level requirements, the responsibilities should be solution-neutral and stated at the system level to avoid making incorrect assumptions about how the responsibilities will be implemented until later in the architecture development process. To do this, most requirements can be translated into statements of responsibilities by removing the first few words of the requirement and retaining just the part of the requirement that states the constraint as illustrated in Table 7.

Table 7: Translating system-level requirements into responsibilities

Req ID	System-Level Requirement	Resp. ID	Responsibility
SLR-19	The FOS must be able to detect all objects and other aircraft in the environment under all DVE conditions at all times [SR-18]	SR-18	Detect all objects and other aircraft in the environment under all DVE conditions at all times [SLR-19]
SLR-23	The FOS must be able to respond quickly enough to avoid violations of minimum separation [SR-22]	SR-22	Respond quickly enough and with appropriate magnitude to select and effect the desired change in flight path under the given environmental conditions [SLR-23, SLR-25]

Responsibilities also do not necessarily have to be related one-to-one to the system-level requirements. In some cases such as SR-22, a responsibility can be associated with multiple system-level requirements and vice versa. It is also important that traceability is included (shown in the square brackets in Table 7) along with each requirement and responsibility to ensure that anyone reviewing this information in the future can determine which requirements each responsibility is derived from and vice versa.

The 24 responsibilities generated for this case study are as follows:

SR-1. Determine if feedback received about the aircraft's mission readiness, state of the aircraft and the airspace is too old [SLR-1]

SR-2. Account for prevailing DVE conditions and their possible effect on feedback sources when making use of feedback information [SLR-2]

SR-3. Validate the inputs or feedback received from other controllers before using that input or feedback to update process models [SLR-3]

SR-4. Confirm that all aspects of the aircraft state match the expected safe departure state before providing actuator movements to depart [SLR-4]

SR-5. Determine if the state of the aircraft changes between the time that it was checked and departure [SLR-5]

SR-6. Confirm that data about the expected safe departure state of the aircraft has been fully specified and received [SLR-6, SLR-9]

SR-7. Ensure that all relevant personnel are aware of any changes to the expected safe departure state of the aircraft [SLR-7]

SR-8. Process all feedback to make a deliberate decision if it is to be ignored/dropped [SLR-8]

SR-9. Ensure that all data needed to determine the state of the aircraft, state of the airspace and the environmental conditions around the aircraft under all DVE conditions is available at all times [SLR-9, SLR-15]

SR-10. Ensure that flight crews are aware of the most updated state of any maintenance tasks on the aircraft [SLR-10]

SR-11. Ensure that only authorized actuator movements are executed [SLR-11]

SR-12. Be able to operate in the expected DVE conditions [SLR-12]

SR-13. Distinguish useful feedback from noise in feedback data [SLR-13]

SR-14. Process sensor data and use it to update its process model sufficiently quickly [SLR-14]

SR-15. Determine if the state of the airspace changes between the time that it was checked and the commencement of a maneuver [SLR-16]

SR-16. Account for the current and future movements of other aircraft in the vicinity when selecting actuator movements [SLR-17]

SR-17. Ensure that aircraft movements are selected such that sufficient reaction time is available if intent or movements of other aircraft are not what was expected, even if no violation of minimum separation is initially expected [SLR-18]

SR-18. Detect all objects and other aircraft in the environment under all DVE conditions at all times [SLR-19]

SR-19. Detect slow or subtle changes in the state of the aircraft under all DVE conditions at all times [SLR-20]

SR-20. Select actuator movements that minimize the risk of violation of minimum separation when information about the state of the airspace is in a degraded condition (e.g. delayed) [SLR-21]

SR-21. Respond quickly and with appropriate magnitude to disturbances to prevent unintended movement of the aircraft [SLR-22]

SR-22. Respond quickly enough and with appropriate magnitude to select and effect the desired change in flight path under the given environmental conditions [SLR-23, SLR-25]

SR-23. Ensure that a viable flight path is always available to avoid any violations of minimum separation [SLR-24]

SR-24. Select a viable flight path that avoids all violations of minimum separation [SLR-26]

4.4.2 Defining Decision-Making Strategies, Refined Control Actions and Required Process Model Parts

Once the responsibilities are defined, the next step is to define the decision-making strategy, refined control actions and process model parts. It is important that the decision-making strategy is defined first before the refined control actions and required process model parts are defined because the decision-making strategy serves as the basis for refining a control action or identifying required process model parts. Table 8 shows how the decision-making strategy, refined control actions and process model parts should be recorded for SR-18 (associated with SLR-19) and SR-22 (associated with SLR-23).

Table 8: Decision-Making Strategy, Refined Control Actions and Process Model Parts for SR-18 and SR-22

Resp. ID	Responsibility	Decision-Making Strategy	Refined Control Action(s)	Required Process Model Contents	Rationale and Assumptions
SR-18	Detect all objects and other aircraft in the environment under all DVE conditions at all times [SLR-19]	Use data from <sensor choices> to measure the position and speed of objects and aircraft in the airspace	N/A	All process model contents in Current Weather Conditions and Current Airspace State	<Rationale for sensor choices based on anticipated objects and DVE conditions and other performance parameters> <Assumptions about minimum object size and DVE conditions>
SR-22	Respond quickly enough and with appropriate magnitude to select and effect the desired change in flight path under the given environmental conditions [SLR-23, SLR-25]	Normal decision making process needed to select actuator movements, however decision making must happen more quickly	Roll Pitch Yaw	Model of Current Weather Conditions • Outside air temperature • Wind speed • Visibility Model of Current Aircraft Navigation State • Air speed • Altitude • Position Model of System Behavior • Guidelines for aircraft handling in DVEs	Assumes the “rules” for selecting actuator movements will be the same whether in DVE conditions or not but those decisions will need to be more responsive to disturbances caused by some DVE conditions than normal

To define the decision-making strategy, the responsibility should be used to decide what decision(s) are required to carry out the responsibility and how the system as a whole should make those decision(s). For example, SR-18 describes the responsibility for detecting all other aircraft and other objects in the airspace under all DVE conditions. The decision-making strategy might therefore be to use a sensor suite to measure the position and speed of objects and aircraft in the airspace. Although this decision-making strategy is currently too abstract to perform a detailed system design with, this level of abstraction is useful at this stage in architecture development because it provides just enough detail to consider refinements to the control actions and determine the process model parts and feedback that are necessary. At later stages in the architecture development process, this decision-making strategy can then be further refined to include a sufficient level of detail to implement it. Examples of decision-making strategies can be found in the third column of Table 8.

Once the decision-making strategy is defined, refined control actions and the process model parts required to support decision-making can be defined. The decision-making strategy can be used to determine what process model parts might be needed to enable effective decision-making. So, for example, in SR-22, the Piloting Controller will need to know the current weather conditions (e.g. wind speed, ambient temperature), the current navigation state of the aircraft (e.g. current airspeed, current altitude, current position) and the model of system behavior (e.g. how much roll, pitch and yaw is required to achieve a desired flight path). The refined control actions can also be defined in a similar manner by considering what the output of the decision-making strategy might be. For example, in SR-22, since the responsibility is to respond quickly enough and with appropriate magnitude to effect a change in flight path, the output of that decision is likely going to be roll, pitch and yaw inputs. Therefore, the original "Actuator Movements" control action defined at the start of the STPA analysis in Section 4.2 can be refined to be "Roll, Pitch and Yaw". Examples of refined control actions and required process model parts can be found in the fourth and fifth columns of Table 8 respectively.

Since the decision-making strategy, refined control actions and process model parts all represent design decisions being made, it is also important that this information be recorded along with the underlying rationale and assumptions to ensure that it these decisions can be more easily understood and reviewed by a reviewer, assessor or system maintainer. Examples of the rationale and assumptions can be found in the last column of Table 8.

4.4.3 Defining Sources of Feedback

After defining the decision-making strategy, refined control actions and required process model parts for each responsibility, it is now possible to collate all the process model parts together so that the size of the whole process model and all its parts can be visualized together. This visualization can then be used to identify the source of the feedback that is necessary to update and maintain each part of the process model. So, for example, for the model of current weather conditions, since weather conditions should be measured using temperature or wind speed sensors that are included as part of the aircraft subsystems, the temperature and wind speed parts of the process model would be updated using feedback of the same name originating from

the aircraft subsystems. As with the other parts of the system-level behavior information, because this represents a design decision being made about the system, this information should be recorded along with the underlying rationale and assumptions. Table 9 shows how feedback information should be recorded, using the Current Navigation State and Current Airspace State parts of the process model as examples.

Table 9: Feedback Information for Current Navigation State and Current Airspace State

Process Model Part	Required Information	Source of Information	Rationale/Assumptions
Current Aircraft Navigation State	Current Air Speed	Aircraft subsystems	As with today's aircraft, these parameters will continue to be measured using sensors on the airframe
	Current Altitude		
	Current Position		
Current Airspace State	IDs, Positions and Speeds of other aircraft and objects	Aircraft Subsystems Reports from other controllers EMS Operations and ATC	As with today's aircraft, these parameters will either be measured using current methods (e.g. radar contact) or may be reported via other communications channels (e.g. verbal radio communication) from other parts of the system
	Time of last feedback measurement	Aircraft Subsystems	This assumes that timestamps associated with sensor data are used

In total, eight process model parts were identified for this case study:

1. Current Aircraft Mission Readiness
2. Current Aircraft System State
3. Current Aircraft Navigation State
4. Expected Safe Departure State
5. Current Weather Conditions
6. Current Airspace State
7. Anticipated Future Airspace State
8. Model of System Behavior

4.4.4 Defining Timing Requirements

Finally, as discussed in Section 3.4, up to this point, the system-level behavior only contains static information about what is required to carry out the identified responsibilities and the temporal aspects of carrying out the responsibilities have not been made explicit. Therefore, the last part of the system-level behavior information is to specify these temporal aspects.

In this case study, before identifying the timing requirements, the identified responsibilities were divided into two types: Feedback Validation Responsibilities and Non-Feedback-Validation Responsibilities. Feedback Validation Responsibilities are responsibilities that involve the validation of feedback prior to using them to update the process model and examples include validating any inputs from other controllers before using them or accounting for the negative effects of DVE conditions on feedback sources (e.g. sensors) prior to using that feedback to update the process model. On the other hand, Non-Feedback-Validation Responsibilities are all other responsibilities that do not involve the validation of feedback and examples include accounting for the intent of other aircraft when selecting actuator movements and selecting actuator movements quickly and with appropriate magnitude to execute the desired flight path. The reason for dividing the responsibilities into these two groups is that the timing requirements for feedback validation responsibilities might be different depending on the type of feedback whereas non-feedback-validation responsibilities do not have this dependency. As such, for Feedback Validation Responsibilities, timing requirements are defined for each pair of responsibilities and process model parts whereas Non-Feedback-Validation Responsibilities are simply defined for each responsibility.

To define timing requirements, the frequency and speed with which the responsibility must be carried out should be considered by answering two key questions:

1. How often does this constraint need to be enforced? (Frequency)
2. How quickly does a decision about this responsibility need to be made? (Speed)

Similar to the other parts of the system-level behavior information, since these timing requirements represent design decisions being made, they should also be recorded along with the underlying assumptions and rationale.

Table 10 shows how timing requirements are defined for Non-Feedback-Validation Responsibilities by answering the 2 questions above using SR-18 (associated with SLR-19) and SR-22 (associated with SLR-23) as examples.

Table 10: Example Timing Requirements for Non-Feedback-Validation Responsibilities

Resp. ID	Responsibility	Timing Requirements	Rationale/Assumptions
SR-18	Detect all objects and other aircraft in the environment under all DVE conditions at all times [SLR-19]	Based on sensor performance characteristics and time needed after detections are made to plan and execute a flight trajectory, it is necessary to update the process model of the current state of the airspace at a rate of <TBD> Hz. This defines how often this responsibility has to be carried out. Based on this update rate, sensor data and any other inputs must be integrated into a coherent picture of the current state of the airspace within <TBD> seconds. This defines how quickly a decision has to be made for this responsibility	This assumes that <TBD> seconds are required after detections are made to plan and execute a trajectory based on those detections.
SR-22	Respond quickly enough and with appropriate magnitude to select and effect the desired change in flight path under the given environmental conditions	Based on worst case detection ranges and the dynamics and handling characteristics of the aircraft, a change in flight path must be decided and executed within <TBD> seconds	This assumes aircraft dynamics and handling characteristics are <TBD> and the worst case detection ranges are <TBD> as defined in <TBD> document

By contrast, Feedback Validation Responsibilities are defined in a matrix consisting of the process model parts written in the rows and the responsibilities written in the columns. Each cell in this matrix therefore defines the timing requirements for a given responsibility for a particular process model part.

Table 11 shows how timing requirements are defined for two Feedback Validation Responsibilities (SR-2 and SR-3) for two process model parts (Current Aircraft Navigation State and Current Weather Conditions).

SR-2: Account for prevailing DVE conditions and their possible effect on feedback sources when making use of feedback information [SLR-2]

SR-3: Validate the inputs or feedback received from other controllers before using that input or feedback to update process models [SLR-3]

Table 11: Example Timing Requirements for Feedback Validation Responsibilities

Process Model Part	Feedback Responsibilities		Rationale/Assumptions
	SR-2	SR-3	
Current Aircraft Navigation State	Since navigation state must be updated every <TBD> seconds, then all navigation-related feedback must be evaluated every <TBD> seconds as well. In addition, the decision as to whether prevailing DVE conditions might have negative impact must be made within <TBD> seconds	When an external input regarding the aircraft's navigation state is received, it must be validated within <TBD> seconds to ensure that navigation state update rate of <TBD> seconds is maintained. External inputs regarding the aircraft's navigation state are expected to be received at a very infrequent rate.	The required navigation state update rate was determined based on <TBD>. The expectation that the rate of receipt of external input for the aircraft navigation state is infrequent assumes that the only external input the Piloting Controller might receive is verbal input from another aircraft or a controller performing a function similar to that of Air Traffic Control
Current Weather Conditions	Since weather conditions must be updated every <TBD> seconds, then all weather-related feedback must be evaluated every <TBD> seconds as well.	External weather inputs are expected to be received every <TBD> sec and must be processed at the same <TBD> rate to keep weather conditions updated every <TBD> seconds	This assumes external weather inputs generated by <TBD> weather reporting systems are received at the aircraft every <TBD> seconds

It is worth noting that because these timing requirements require decisions in other parts of the system-level behavior information to have been made (e.g. choice of sensors), some information described in the timing requirements in Table 10, Table 11 and Appendix D are written with "<TBD>" used as a placeholder. Eventually, the "<TBD>" placeholders should be replaced with the required information or numerical values once the necessary decisions have been made.

Once these timing requirements have been defined, this concludes the definition of system-level behavior.

Intent Specification Level 2: System Engineering View

4.5 Step 4, Part 1: Creating and Assessing Architecture Options for the Overall System

As described in Section 3.5, the first part of the 2-part architecture creation process involves creating and assessing architecture options for the overall system (i.e. the FOS). First, an overview of the architecture options is provided in Section 4.5.1. Then, the assignments for each architecture option are shown and each architecture is analyzed individually in Sections 4.5.2 to 4.5.5. Finally, in Section 4.5.6, the two architecture options are compared and the benefits and tradeoffs between them are discussed to conclude part 1 of the architecture creation process.

4.5.1 Overview of Architecture Options

In this section, two architecture options will be presented for the FOS:

- Option 1: The Existing Flight Operations System (FOS)
- Option 2: The Enhanced FOS

Option 1 represents the simplest system for executing a mission where Maintenance personnel maintain the aircraft, EMS Operations and ATC assist with planning the mission and ensuring the aircraft has access to the airspace and the piloting controller is primarily responsible for all aspects of executing the mission and safely flying the aircraft. A key aspect of this architecture option is that the piloting controller detects and updates the current and future state of the airspace using only a-priori information from maps and other databases available to emergency services personnel and assumes limited ATC support (e.g. in uncontrolled airspace).

By contrast, architecture option 2 is a modified version of option 1 where EMS Operations and ATC can assist with maintaining and updating the current and future state of the airspace in which the mission is being executed. For example, EMS Operations and ATC could take advantage of data or verbal reports from other flight crews or other emergency responders who might also be responding to the scene of an emergency to assist the Piloting Controller in maintaining and updating a shared understanding of the locations of hazards such as cell towers, terrain features and other objects and hazards. As such, the Piloting Controller's ability to maintain an updated model of the current and future state of the airspace is enhanced because EMS Operations and ATC can assist the Piloting Controller in maintaining an updated model of the current and future state of the airspace by incorporating additional information that the Piloting Controller would not otherwise have access to.

For all the responsibility and process model assignments, the names of the controllers they are assigned to will be abbreviated as follows:

- EMSATC: EMS Operations and ATC
- MP: Maintenance Personnel
- PC: Piloting Controller
- AS: Aircraft Subsystems

4.5.2 Architecture Option 1: The Conventional Flight Operations System

As described in the overview, this architecture option represents an architecture for the FOS where the Piloting Controller is primarily responsible for flying the aircraft and executing the mission. As a result, most of the process model parts and responsibilities are assigned to the Piloting Controller and only a few are shared with other controllers in the system. The assignment of process model parts will be shown first followed by the assignment of non-feedback-validation responsibilities and finally the assignment of feedback validation responsibilities. While only examples of the responsibility assignments are shown in this section, full details for this architecture option can be found in Appendix E.

Assignment of Process Model Parts

The assignments for each of the eight process model parts identified in the system-level behavior information are shown in Table 12 along with the underlying rationale and assumptions for each assignment.

Table 12: Assignments of Process Model Parts for Part 1 Architecture Option 1

Process Model	Assigned to				Rationale/Assumptions
	EMSATC	MP	PC	AS	
Current Aircraft Mission Readiness		X	X		Since maintenance personnel will prepare and maintain the aircraft on behalf of the piloting controller, this process model is shared between them
Current Aircraft System State			X	X	Since the aircraft subsystems assist the piloting controller in monitoring the state of the aircraft subsystems (e.g. engine status, hydraulic system status), the aircraft system state is shared between the piloting controller and aircraft subsystems
Current Aircraft Navigation State	X		X		Since EMS Operations and ATC is likely monitoring the progress of the mission, the aircraft navigation state will be shared with it and the Piloting Controller.
Expected Safe Departure State	X	X	X		For any mission, EMS Operations and ATC establishes the mission plan and the maintenance personnel will determine the maintenance state of the aircraft when it will be needed for the mission. All of this information will also be needed by the piloting controller to perform pre-flight and departure checks. As a result, this information is shared by all three controllers

Process Model	Assigned to				Rationale/Assumptions
	EMSATC	MP	PC	AS	
Current Weather Conditions			X		Since these process model parts are used and updated constantly in real-time during a flight, the piloting controller must be assigned all of them. In addition, it is assumed in this architecture option that no other controller is equipped to assist the piloting controller in maintaining these process model parts
Current Airspace State			X		
Anticipated Future Airspace State			X		
Model of System Behavior	X		X	X	Not only does EMS Operations and ATC set some of the parameters based on aircraft acquired and other factors, EMS Operations and ATC must also use some of the parts of this process model part to plan the mission. The aircraft subsystems and piloting controller must also know about some of these parts of the process model to control the aircraft and execute a mission

For this architecture option, all the process model parts are assigned to the Piloting Controller because they are primarily responsible for flying the aircraft and will therefore need all of the process model parts at some point during the flight to carry out all the necessary responsibilities. In addition, while most of the process model parts are only assigned to the Piloting Controller because they need to be carried out continuously during flight, some process model parts are shared with other controllers in the FOS. For example, the Current Aircraft Mission Readiness process model part is shared with the maintenance personnel because the maintenance personnel need to know the state of the aircraft to ensure it is ready for a mission but the Piloting Controller also needs to have this process model part so that it knows when the aircraft is ready to depart on the mission. Similarly, the expected safe departure state of the aircraft is shared with EMS Operations and ATC who plan the mission and manage the airspace, the Maintenance personnel who need to know how the aircraft should be configured for the mission and the Piloting Controller who needs to know what state their aircraft should be in before departing.

Assignment of Non-Feedback-Validation Responsibilities

As described in Section 4.4.4, the non-feedback-validation responsibilities are a subset of the responsibilities identified for the FOS that do not describe constraints on how feedback should be validated before it is used to update the process model parts. As might be expected, since the Piloting Controller is primarily responsible for flying the aircraft in this architecture option, most of the responsibilities related to safe flight are assigned only to the Piloting Controller and some

examples are shown in Table 13. This is because only the Piloting Controller has direct access to the required feedback (e.g. sensor and detection data) defined in the system-level behavior to be able to carry out these responsibilities at the required speed and frequency defined in the timing requirements of the system-level behavior.

Table 13: Flight-Related Non-Feedback-Validation Responsibilities Assigned Only to the PC in Part 1 Architecture Option 1

Resp. ID	Responsibility	Assigned to				Rationale/Assumptions
		EMSATC	MP	PC	AS	
SR-4	Confirm that all aspects of the aircraft state match the expected safe departure state before providing actuator movements to depart [SLR-4]			X		Since the piloting controller is primarily responsible for controlling the aircraft and executing the mission, this related responsibility for making final confirmation that the aircraft is in a safe departure state before departing should also be assigned to the piloting controller
SR-15	Determine if the state of the airspace changes between the time that it was checked and the commencement of a maneuver [SLR-16]			X		Since the piloting controller is primarily responsible for controlling the aircraft and this responsibility relates to how to maintain safe control of the aircraft, it is assigned to the piloting controller
SR-16	Account for the current and future movements of other aircraft in the vicinity when selecting actuator movements [SLR-17]			X		All of these responsibilities relate to selecting appropriate flight paths and actuator movements, all of which are within the scope of the piloting controller's responsibility for maintaining safe control over the aircraft. As such, these responsibilities are all assigned to the piloting controller
SR-22	Respond quickly enough and with appropriate magnitude to select and effect the desired change in flight path under the given environmental conditions [SLR-23, SLR-25]			X		
SR-24	Select a viable flight path that avoids all possible violations of minimum separation [SLR-26]			X		

By contrast, some responsibilities are either shared between the Piloting Controller and other controllers in the FOS or not assigned to the piloting controller at all. Examples of these assignments for this architecture option are shown in Table 14.

Table 14: Part 1 Architecture Option 1 Assignment of Other Non-Feedback-Validation Responsibilities

Resp. ID	Responsibility	Assigned to				Rationale/Assumptions
		EMSATC	MP	PC	AS	
SR-6	Confirm that data about the expected safe departure state of the aircraft has been fully specified and received [SLR-6, SLR-9]	X		X		Since the expected safe departure state is dependent on the needs of a specific mission, EMS Operations and ATC must be assigned responsibility for planning the mission. However, the piloting controller could also confirm that the information it receives is complete
SR-7	Ensure that all relevant personnel are aware of any changes to the expected safe departure state of the aircraft [SLR-7]	X				Assuming that changes are likely to either come from EMS Operations and ATC or need to be approved by EMS Operations and ATC, then EMS Operations and ATC will be informed of any changes that are needed and therefore is in the best position to disseminate any changes should assigned to EMS Operations and ATC
SR-11	Ensure that only authorized actuator movements are executed [SLR-11]				X	Since the execution of actuator movements to move actuators is performed by the aircraft subsystems, the responsibility for only executing authorized actuator movements is assigned to the aircraft subsystems
SR-18	Detect all objects and other aircraft in the environment under all DVE conditions at all times [SLR-19]			X	X	Under DVE conditions, the piloting controller's direct observations will need to be augmented by a sensor suite that is part of the aircraft subsystems

Some responsibilities such as SR-7 and SR-11 are not assigned to the Piloting Controller at all because they describe constraints that are more easily enforced by another controller in the FOS

instead of the Piloting Controller. For example, SR-11 is only assigned to the aircraft subsystems because the aircraft subsystems interact directly with the physical actuators and hardware on the aircraft and the piloting controller does not. On the other hand, some responsibilities such as SR-6 and SR-18 are shared because both the Piloting Controller as well as the other controller can play a role in carrying out that responsibility For example, SR-18 is assigned to both the piloting controller and aircraft subsystems in this architecture option because the aircraft subsystems contain the sensors which perform the data collection and some basic interpretation of that sensor data to produce detections and the piloting controller integrates all the individual detections and resolves any conflicting feedback to produce a single coherent set of detections.

Assignment of Feedback Validation Responsibilities

Finally, the feedback validation responsibilities also need to be assigned to the controllers in the FOS. As described in Section 4.4.4, the feedback validation responsibilities are the subset of the responsibilities identified for the FOS that describe constraints on how feedback should be validated before it is used to update the process model parts. As such, for these responsibilities, an assignment is made for each pair of process model part and responsibility. So, for example, a separate assignment is defined for SR-1 for each of the eight process model parts.

Similar to the non-feedback-validation responsibilities, since the piloting controller is primarily responsible for integrating the various sources of feedback to fly the aircraft safely, all of the feedback responsibilities for almost every process model part is assigned to the piloting controller in this architecture option. In some cases, however, no assignment is made because the responsibility is not expected to be applicable for that process model part. For example, although it is necessary to account for the effects of DVE when making use of feedback about the current state of the airspace, it is not necessary to do so when making use of feedback about the expected safe departure state because the type of feedback that will be received is not expected to be affected by DVE. In instances like these, “N/A” is used to indicate that no assignment is made and the rationale for why the responsibility is not applicable for that process model part is described in the rationale/assumptions column. Some examples of feedback validation responsibility assignments for this architecture option are shown in Table 15.

Table 15: Assignment of Feedback Validation Responsibilities for Part 1 Architecture Option 1

<table>
<tr><th>Resp ID</th><th>Responsibility</th><th>Assignments for Each Process Model Part</th><th>Rationale/Assumptions</th></tr>
<tr><td>SR-2</td><td>Account for prevailing DVE conditions and their possible effect on feedback sources when making use of feedback information [SLR-2]</td><td><table>
<tr><td>Aircraft Mission Readiness</td><td>N/A</td></tr>
<tr><td>Current Aircraft System State</td><td>PC</td></tr>
<tr><td>Current Aircraft Navigation State</td><td>PC</td></tr>
<tr><td>Expected Safe Departure State</td><td>N/A</td></tr>
<tr><td>Current Weather Conditions</td><td>PC</td></tr>
<tr><td>Current Airspace State</td><td>PC</td></tr>
<tr><td>Anticipated Future Airspace State</td><td>PC</td></tr>
<tr><td>Model of System Behavior</td><td>N/A</td></tr>
</table></td><td>Since this responsibility depends on prevailing weather conditions during the flight, this responsibility must be carried repeatedly and in real-time. As such, the piloting controller is the only controller equipped to do this.

This assumes that feedback for aircraft mission readiness, expected safe departure state and model of system behavior will not be affected by DVE and hence no assignment is made for these parts</td></tr>
<tr><td>SR-3</td><td>Validate the inputs or feedback received from other controllers before using that input or feedback to update process models [SLR-3]</td><td><table>
<tr><td>Aircraft Mission Readiness</td><td>PC</td></tr>
<tr><td>Current Aircraft System State</td><td>PC</td></tr>
<tr><td>Current Aircraft Navigation State</td><td>PC</td></tr>
<tr><td>Expected Safe Departure State</td><td>EMSATC</td></tr>
<tr><td>Current Weather Conditions</td><td>PC</td></tr>
<tr><td>Current Airspace State</td><td>PC</td></tr>
<tr><td>Anticipated Future Airspace State</td><td>PC</td></tr>
<tr><td>Model of System Behavior</td><td>EMSATC</td></tr>
</table></td><td>Similar to SR-1, the piloting controller is in the best position to decide when feedback is too old since these process model parts are used to fly the aircraft or select a flight path.

This includes the model of system behavior which could have its parameters updated based on input that needs to be validated</td></tr>
</table>

<table>
<tr><th>Resp ID</th><th>Responsibility</th><th>Assignments for Each Process Model Part</th><th>Rationale/Assumptions</th></tr>
<tr><td>SR-9</td><td>Ensure that all data needed to determine the state of the aircraft, state of the airspace and the environmental conditions around the aircraft under all DVE conditions is available at all times [SLR-9, SLR-15]</td><td><table>
<tr><td>Aircraft Mission Readiness</td><td>PC</td></tr>
<tr><td>Current Aircraft System State</td><td>PC</td></tr>
<tr><td>Current Aircraft Navigation State</td><td>PC</td></tr>
<tr><td>Expected Safe Departure State</td><td>N/A</td></tr>
<tr><td>Current Weather Conditions</td><td>PC</td></tr>
<tr><td>Current Airspace State</td><td>PC</td></tr>
<tr><td>Anticipated Future Airspace State</td><td>PC</td></tr>
<tr><td>Model of System Behavior</td><td>N/A</td></tr>
</table></td><td rowspan="2">Similar to SR-2, these responsibilities have to be performed repeatedly during flight as described in the timing requirements section. As such, the piloting controller is the only controller in the system that is equipped to do this.

These assignments assume that continuous feedback is not expected for the expected safe departure state and model of system and hence this responsibility does not apply for those process model parts.</td></tr>
<tr><td>SR-13</td><td>Distinguish useful feedback from noise in feedback data [SLR-13]</td><td><table>
<tr><td>Aircraft Mission Readiness</td><td>PC</td></tr>
<tr><td>Current Aircraft System State</td><td>PC</td></tr>
<tr><td>Current Aircraft Navigation State</td><td>PC</td></tr>
<tr><td>Expected Safe Departure State</td><td>N/A</td></tr>
<tr><td>Current Weather Conditions</td><td>PC</td></tr>
<tr><td>Current Airspace State</td><td>PC</td></tr>
<tr><td>Anticipated Future Airspace State</td><td>PC</td></tr>
<tr><td>Model of System Behavior</td><td>N/A</td></tr>
</table></td></tr>
<tr><td>SR-14</td><td>Process sensor data and use it to update its process model sufficiently quickly [SLR-14]</td><td><table>
<tr><td>Aircraft Mission Readiness</td><td>PC</td></tr>
<tr><td>Current Aircraft System State</td><td>PC</td></tr>
<tr><td>Current Aircraft Navigation State</td><td>PC</td></tr>
<tr><td>Expected Safe Departure State</td><td>EMSATC</td></tr>
<tr><td>Current Weather Conditions</td><td>PC</td></tr>
<tr><td>Current Airspace State</td><td>PC</td></tr>
<tr><td>Anticipated Future Airspace State</td><td>PC</td></tr>
<tr><td>Model of System Behavior</td><td>EMSATC</td></tr>
</table></td><td>Similar to SR-2, these responsibilities have to be performed repeatedly during flight as described in the timing requirements section. As such, the piloting controller is the only controller in the system that is equipped to do this.

These assignments assume that continuous feedback is not expected for the expected safe departure state and model of system and hence this responsibility does not apply for those process model parts.</td></tr>
</table>

Architecture Option 1 Control Structure

The resulting control structure for this architecture option is shown in Figure 19.

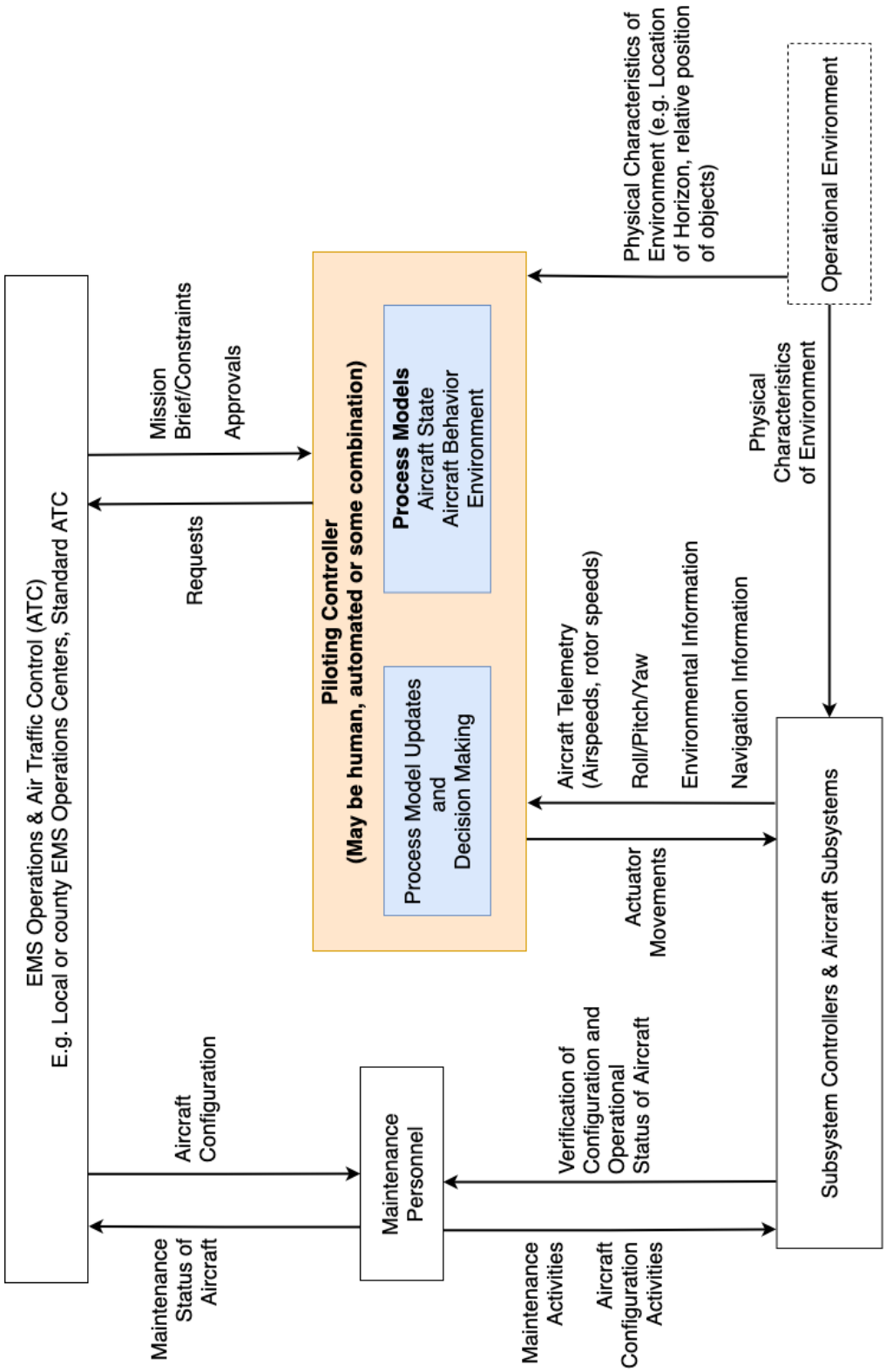


Figure 19: Part 1 Architecture Option 1 Control Structure

4.5.3 Architecture Analysis of Option 1

There are two key observations that can be made about this architecture option that could inform architectural decision making. The first is that this architecture option imposes high workload and capability requirements on the Piloting Controller. Since the Piloting Controller carries out most of the responsibilities related to safe flight in this architecture option, most of the additional DVE-related requirements are assigned to it. As a result, the workload and capability requirements imposed on the Piloting Controller are significantly increased compared to flying in non-DVE conditions because it must now meet these new requirements. Examples of loss scenarios and the associated responsibilities that are assigned to the Piloting Controller that illustrate the increased workload or capability requirements are shown in Table 16.

Table 16: Scenarios and Responsibilities Showing Increased Workload or Capability Requirements

Scenario ID	Scenario	Resp. ID	Responsibility
CS-1.1.1-1.1.2	The piloting controller believes it is receiving conflicting feedback because DVE conditions have degraded the accuracy of or obscured some (but not all) feedback sources. The piloting controller therefore chooses to ignore the conflicting feedback, believing that it is a false positive or erroneous. The piloting controller is especially susceptible to this if the majority of sources appear to agree that no aircraft is present and fewer sources appear to show otherwise. As a result, the piloting controller updates its process model based only on what it wrongly believes to be the correct feedback. [SLR-2]	SR-2	Account for prevailing DVE conditions and their possible effect on feedback sources when making use of feedback information [SLR-2]
CS-1.2.1-1.1	DVE conditions cause more noise in sensor data than is normally present, making the useful feedback more difficult for the piloting controller to distinguish from the noise or increasing the likelihood that the piloting controller wrongly decides that the sensor data shows no useful feedback/detections. [SLR-13, SLR-19]	SR-13	Distinguish useful detections/feedback from the noise that might be present in feedback data under all DVE conditions at all times [SLR-13]
CS-1.5.4-1	The piloting controller may have the wrong process model of the environment conditions and selects actuator movements that are insufficient to effect the desired change in flight path [SLR-25]	SR-22	Respond quickly enough and with appropriate magnitude to select and effect the desired change in flight path under the given environmental conditions [SLR-23, SLR-25]

SR-2 is an example of an additional responsibility that is imposed on the Piloting Controller because it is flying the aircraft under DVE conditions. If DVE conditions were not present, the Piloting Controller would not need to account for DVE conditions when making use of feedback. By contrast, SR-13 and SR-22 are examples of existing responsibilities that are made more difficult to carry out due to the presence of DVE conditions. Even in the absence of DVE conditions, the Piloting Controller would still be required to distinguish useful feedback from noise and select actuator movements that are sufficient and appropriate to effect the desired change in the flight path. However, due to the presence of DVE conditions, these requirements are harder to satisfy because DVE conditions might decrease the signal to noise ratio for some sensors and require the Piloting Controller to consider a larger number of factors in a shorter amount of time when selecting actuator movements. As such, these responsibilities illustrate how the presence of DVE conditions increases the workload and capability requirements of the Piloting Controller by either making existing requirements harder to satisfy or imposing new additional requirements that the Piloting Controller must satisfy.

The second key observation is that this architecture option imposes stringent performance and feedback requirements that may be difficult or impossible to meet. As a result, there is an increased risk that architectural decisions may need to be changed if it is later found during detailed system design that the system requirements cannot be met. Since most of the feedback validation responsibilities are assigned to the Piloting Controller and the Piloting Controller is reliant on sensor data to detect objects and other aircraft in the airspace, the performance of the Aircraft Subsystems and the ability of the Piloting Controller to use that feedback to maintain an updated process model of the state of the airspace is extremely critical to ensuring the safe operation of the FOS. If the Piloting Controller or the feedback that it relies on from the Aircraft Subsystems is unable to meet the stringent requirements, the Piloting Controller will be vulnerable to unsafe behavior during flight in DVE conditions. Consequently, if the stringent performance and feedback requirements are difficult or impossible to meet, not only could the safety of this architecture be compromised but there is also an increased risk that architectural decisions may need to be changed to make them achievable. Table 17 contains three scenarios along with the associated responsibilities that are assigned to the Piloting Controller that illustrate examples of the performance and quality requirements that must be met.

Table 17: Scenarios Suggesting Stringent Performance and Quality Requirements

Scenario ID	Scenario	Resp. ID	Responsibility
CS-1.2.1-1.1	DVE conditions cause more noise in sensor data than is normally present, making the useful feedback more difficult for the piloting controller to distinguish from the noise or increasing the likelihood that the piloting controller wrongly decides that the sensor data shows no useful feedback/detections. [SLR-13, SLR-19]	SR-13	Distinguish useful feedback from noise in feedback data [SLR-13]
CS-1.2.1-1.3	DVE causes delays in the piloting controller interpreting the sensor data and using the sensor data to update its process model. As a result, the piloting controller has the wrong process model of the airspace around the aircraft until it is able to update its process model. [SLR-14]	SR-14	Process sensor data and use it to update its process model sufficiently quickly [SLR-14]
CS-1.2.2-1	DVE degrades or obscures sensors, leading to wrong, incomplete or missing feedback about the environment, DVE conditions or the state of the aircraft. This could also be caused by a sensor suite that was not designed to operate in a particular set of DVE conditions [SLR-2, SLR-15, SLR-19]	SR-2	Account for prevailing DVE conditions and their possible effect on feedback sources when making use of feedback information [SLR-2]
		SR-9	Ensure that all data needed to determine the state of the aircraft, state of the airspace and the environmental conditions around the aircraft under all DVE conditions is available at all times [SLR-9, SLR-15]
		SR-18	Detect all objects and other aircraft in the environment under all DVE conditions at all times [SLR-19]

By combining the responsibilities listed in Table 17 with the system-level behavior defined for each responsibility, the requirements imposed on each component of the system to enable the Piloting Controller to carry out these responsibilities are illustrated in Figure 20.

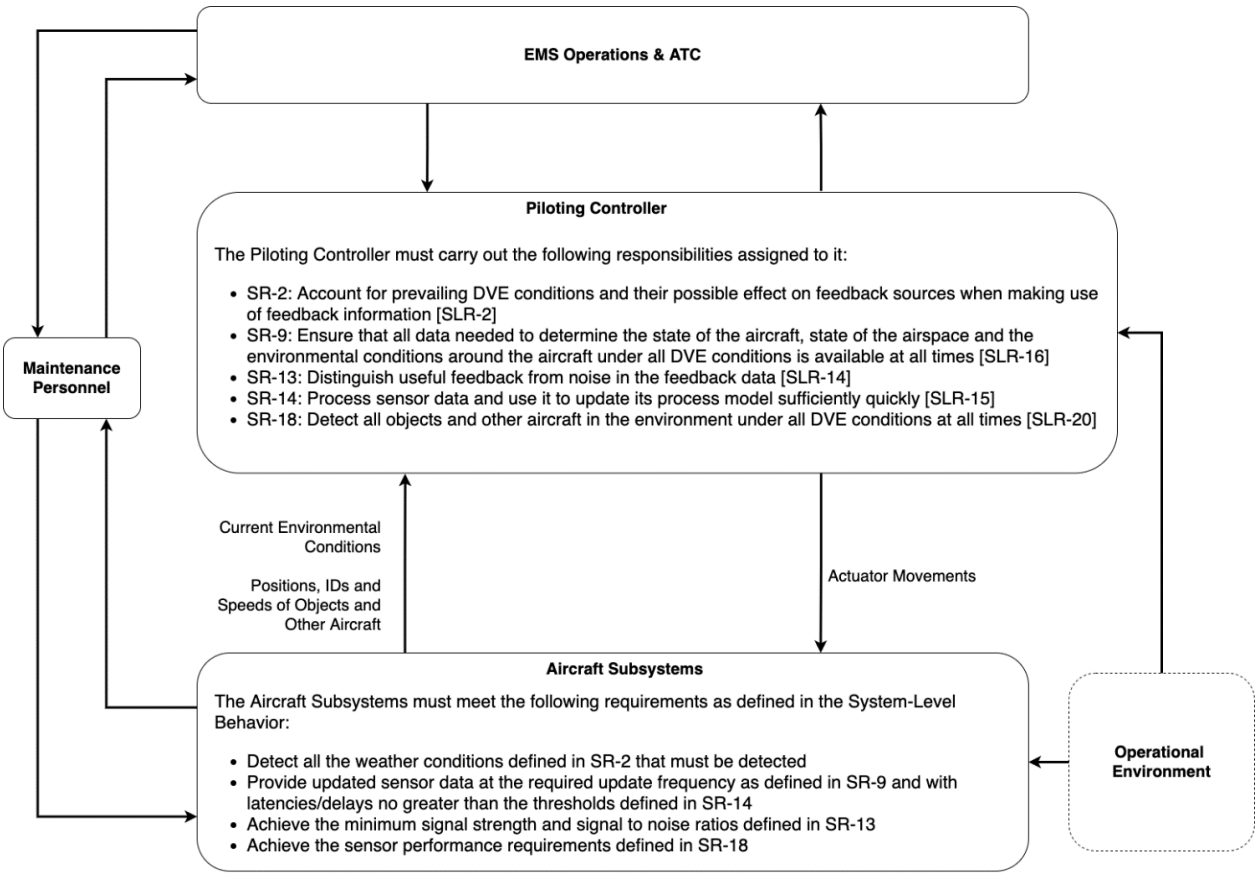


Figure 20: Performance Requirements for Each Part of the System

As shown in the "Piloting Controller" box in Figure 20, the Piloting Controller is responsible for carrying out responsibilities SR-2, SR-9, SR-13, SR-14 and SR-18 as assigned in this architecture option. Using the system-level behavior information defined for these responsibilities, three key aspects must be implemented in the system design. Firstly, to carry out these responsibilities effectively, the Piloting Controller must receive feedback about the current weather conditions and the positions, IDs and speeds of other aircraft and objects in the airspace so that it can keep its process model parts updated. Secondly, the Piloting Controller should receive all that feedback from the Aircraft Subsystems. Finally, that feedback needs to meet the expected performance and quality requirements defined in the system-level behavior in order for the Piloting Controller to receive adequate feedback to carry out its assigned responsibilities effectively. As listed in the "Aircraft Subsystems" box in Figure 20, examples of the performance and quality requirements that the Aircraft Subsystems must meet to provide adequate feedback to the Piloting Controller include:

- Detecting all the weather conditions defined in SR-2
- Providing updated sensor data at the required update frequency as defined in SR-9 and with latencies/delays no greater than the thresholds defined in SR-14
- Achieve the minimum signal strength and signal to noise ratios defined in SR-13
- Achieve the sensor performance (e.g. resolution, range) as defined in SR-18

This therefore shows that, for the four scenarios shown in Table 17 to be adequately mitigated or prevented, the Piloting Controller and Aircraft Subsystems need to meet all the performance and feedback requirements shown in the above diagram.

In this architecture option, however, it may be difficult for the Piloting Controller and Aircraft Subsystems to fully meet these requirements under all conditions since the Piloting Controller is solely responsible for carrying out the five system-level requirements described above using only the sensors on the aircraft as its primary feedback source. For example, if not carefully designed, the Aircraft Subsystems might not be able to sense power lines reliably in DVE conditions because they may be too small to be detected by candidate sensors under DVE conditions. In addition, it may be very expensive or difficult to integrate sensors that could perform all the required detections. As such, by relying solely on the Aircraft Subsystems and Piloting Controller to detect and maintain an updated state of the airspace, this architecture places stringent requirements that must be met by the Aircraft Subsystems and Piloting Controller. If it is discovered during detailed system design that the stringent requirements cannot be met, however, architectural decisions may need to be changed.

4.5.4 Architecture Option 2: The Enhanced Flight Operations System

An alternative architecture to that shown in option 1 is an architecture that provides the Piloting Controller with access to additional information from other sources that it can use to maintain its process model of the current and future state of the airspace. This would allow the Piloting Controller to avoid being entirely dependent on real-time sensing of the environment. One possible way to do this is to create a common situational database that can be updated based on reports from other first responders or other air crews responding to the emergency scene. This type of architecture will therefore be analyzed in this architecture option.

For compactness, only the process model parts and responsibilities that have different assignments than in architecture option 1 will be shown. Any process model part or responsibility that has the same assignment in both architecture options will not be shown again. For all the responsibility and process model assignments, the names of the controllers they are assigned to will be abbreviated as follows:

- EMSATC: EMS Operations and ATC
- MP: Maintenance Personnel
- PC: Piloting Controller
- AS: Aircraft Subsystems

Assignment of Process Model Parts

In this architecture option, only two of the process model parts have different assignments compared to option 1. For the Current Airspace State and Anticipated Future Airspace State, instead of only being assigned to the Piloting Controller, these two process model parts are now shared with EMS Operations and ATC as shown in Table 18. These process model parts are shared because EMS Operations and ATC assist the Piloting Controller in maintaining and updating these parts of the process model in this architecture option and thus need these process model parts assigned to them.

Table 18: Modified Assignments of Process Model Parts in Part 1 Architecture Option 2

Process Model	Assigned to				Rationale/Assumptions
	EMSATC	MP	PC	AS	
Current Airspace State	X		X		In this option, EMS operations and ATC are equipped to assist the piloting controller in maintaining the current and future state of the airspace in which the mission will be taking place. As a result, they share these process model parts.
Anticipated Future Airspace State	X		X		

Assignment of Non-Feedback-Validation Responsibilities

For the assignment of non-feedback-validation responsibilities, two responsibilities have different assignments in this architecture option compared to architecture option 1 and they are shown in Table 19.

For SR-4, the Maintenance personnel are now assigned to help the Piloting Controller to confirm that the aircraft state matches the expected safe departure state instead of requiring the Piloting Controller to perform this responsibility on their own.

For SR-18, EMS Operations and ATC now share this responsibility and can assist the Piloting Controller in identifying other aircraft or objects in the airspace and anticipating what the future state of the airspace might be. For example, EMS Operations and ATC might have access to radar data of the airspace near the emergency scene or video footage of the emergency scene that the Piloting Controller would not have access to. As a result, EMS Operations and ATC can provide updates to the shared process model of the current or anticipated future airspace state for the Piloting Controller.

Table 19: Modified Assignments of Non-Feedback-Validation Responsibilities for Part 1 Architecture Option 2

Resp. ID	Responsibility	Assigned to				Rationale/Assumptions
		EMSATC	MP	PC	AS	
SR-4	Confirm that all aspects of the aircraft state match the expected safe departure state before providing actuator movements to depart [SLR-4]		X	X		Instead of the Piloting Controller being the only controller assigned this responsibility, the maintenance personnel can assist the piloting controller in confirming that the aircraft is configured and that the necessary maintenance is performed
SR-18	Detect all objects and other aircraft in the environment under all DVE conditions at all times [SLR-19]	X		X		Since the EMSATC personnel are assisting in providing the Piloting Controller with information about the current and anticipated future state of the airspace, then the EMSATC personnel share this responsibility with the PC

Assignment of Feedback Validation Responsibilities

Similarly, for the assignment of feedback validation responsibilities, only two responsibilities have different assignments in this architecture option compared to architecture option 1 and they are shown in Table 20.

For SR-1, when determining if some feedback is too old, EMS Operations and ATC shares that responsibility because they are better positioned to assess and validate some types of feedback. For example, when deciding whether feedback about hazards in the vicinity of the emergency scene should be used to update the process model of the state of the airspace, EMS Operations and ATC may be better able to decide when that feedback should be considered out of date. For SR-9, EMS Operations and ATC may be better equipped than the Piloting Controller to detect some objects and aircraft in the airspace and can relay that information to the Piloting Controller. For these reasons, EMS Operations and ATC can share these responsibilities and assist the Piloting Controller in updating their process model of the current and anticipated future airspace state by using and validating additional sources of feedback.

Table 20: Modified Assignments of Feedback Validation Responsibilities for Part 1 Architecture Option 2

<table>
<tr><th>Resp ID</th><th>Responsibility</th><th>Assignments for Each Process Model Part</th><th>Rationale/Assumptions</th></tr>
<tr><td>SR-1</td><td>Determine if feedback received about the aircraft's mission readiness, state of the aircraft and the airspace is too old [SLR-1]</td><td><table>
<tr><td>Aircraft Mission Readiness</td><td>PC</td></tr>
<tr><td>Current Aircraft System State</td><td>PC</td></tr>
<tr><td>Current Aircraft Navigation State</td><td>PC</td></tr>
<tr><td>Expected Safe Departure State</td><td>PC</td></tr>
<tr><td>Current Weather Conditions</td><td>PC</td></tr>
<tr><td>Current Airspace State</td><td>PC
EMSATC</td></tr>
<tr><td>Anticipated Future Airspace State</td><td>PC
EMSATC</td></tr>
<tr><td>Model of System Behavior</td><td>EMSATC</td></tr>
</table></td><td>For all parts except current and anticipated future airspace state, the rationale is the same as in option 1.

For current airspace state and anticipated future airspace state, EMS Operations and ATC is equipped to assist the Piloting Controller in determining if feedback about the current and future airspace state is too old. This is needed because EMS Operations and ATC has access to other data sources that the piloting controller does not have that can help when direct observation is hindered by DVE conditions</td></tr>
<tr><td>SR-9</td><td>Ensure that all data needed to determine the state of the aircraft, state of the airspace and the environmental conditions around the aircraft under all DVE conditions is available at all times [SLR-9, SLR-15]</td><td><table>
<tr><td>Aircraft Mission Readiness</td><td>PC</td></tr>
<tr><td>Current Aircraft System State</td><td>PC</td></tr>
<tr><td>Current Aircraft Navigation State</td><td>PC</td></tr>
<tr><td>Expected Safe Departure State</td><td>N/A</td></tr>
<tr><td>Current Weather Conditions</td><td>PC</td></tr>
<tr><td>Current Airspace State</td><td>EMSATC
PC</td></tr>
<tr><td>Anticipated Future Airspace State</td><td>EMSATC
PC</td></tr>
<tr><td>Model of System Behavior</td><td>N/A</td></tr>
</table></td><td>Similar to SR-1, EMS Operations and ATC can also help the Piloting Controller to ensure that some of the data sources necessary to maintain the models of the current and future airspace state are always available.</td></tr>
</table>

Architecture Option 2 Control Structure

The resulting control structure for this architecture option is shown in Figure 21.

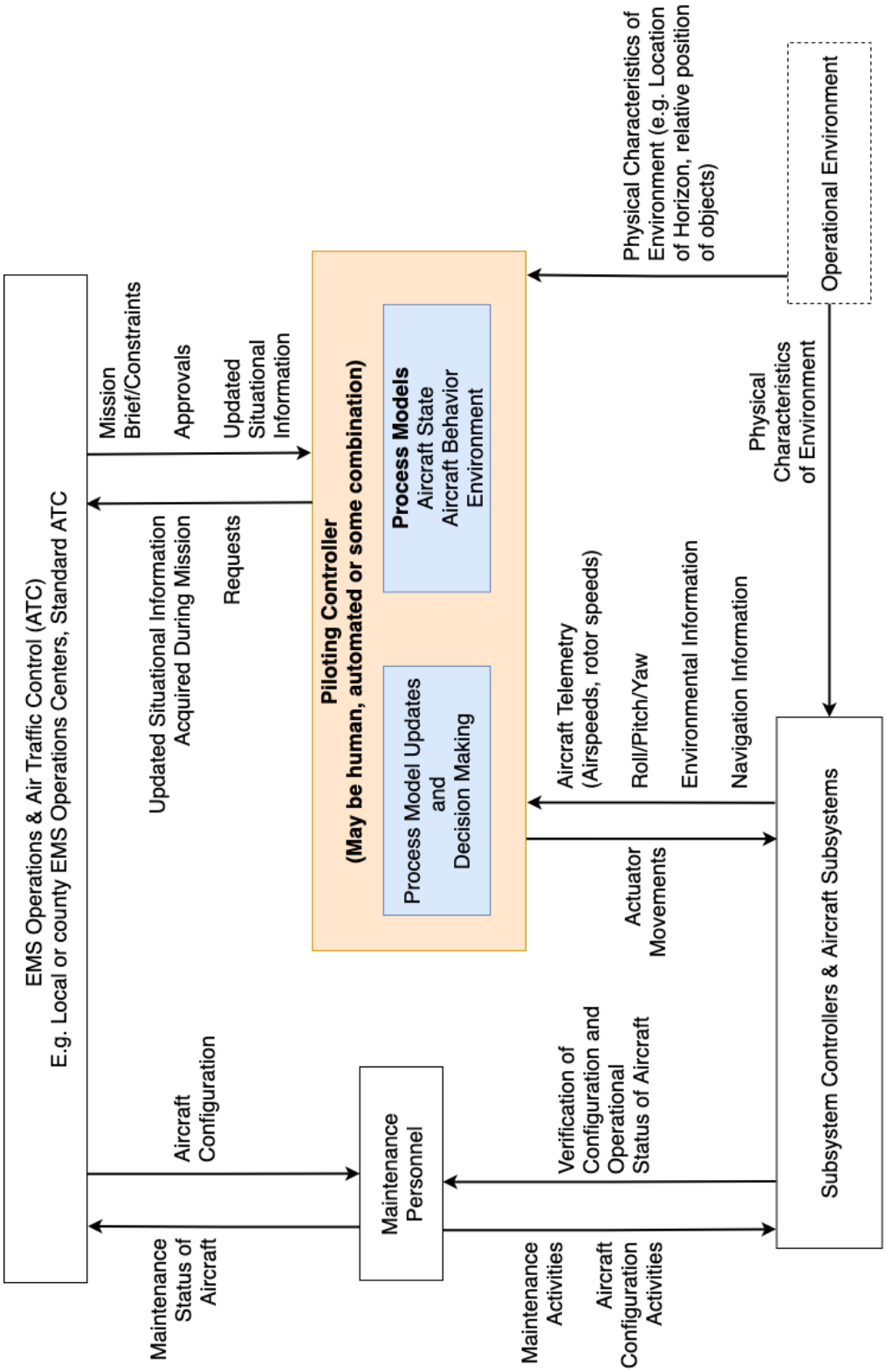


Figure 21: Part 1 Architecture Option 2 Control Structure

4.5.5 Architecture Analysis of Option 2

For this architecture option, several of the scenarios identified in the STPA analysis conducted in Section 3 need to be updated to reflect the responsibilities and process model parts that are now shared in this architecture option. Table 21 shows two example loss scenarios that need to be updated.

Table 21: Examples of Scenarios to be Updated for Architecture Option 2

Scenario ID	Scenario
CS-1.2.1-1.2.2	The piloting controller believes it is receiving conflicting feedback because DVE conditions have degraded the accuracy of or obscured some (but not all) feedback sources. The piloting controller therefore chooses to ignore the conflicting feedback, believing it is a false positive or erroneous
CS-1.2.1-1.4	If another aircraft or controller external to the aircraft providing airspace guidance is in contact with the aircraft and wrongly believes that the airspace nearby the aircraft is clear (for any of the same reasons above) or if a-priori data indicates that no ground obstacles are present, the piloting controller may assume that the external controller/a-priori data is providing accurate information and disregard the onboard sensors in favor of the guidance provided by the external controller. If the piloting controller is human, this may also contribute to confirmation bias, causing the piloting controller not to look for any other evidence that another aircraft or object is nearby. As a result, the piloting controller wrongly updates its process model of the airspace nearby the aircraft [SLR-3, SLR-15]

For CS-1.2.1-1.2.2, now that the process model of the current airspace state is shared between EMS Operations and ATC and the Piloting Controller, there are more ways that this scenario could occur because the shared process model itself can be an additional source of potential conflicts. Not only could the Piloting Controller receive conflicting feedback due to DVE conditions degrading the feedback sources but the shared process model itself could now also conflict with the feedback. This scenario therefore needs to be updated to include the shared process model between EMS Operations and ATC and the Piloting Controller as a potential source of conflicting feedback that the Piloting Controller will need to resolve.

Similarly, for CS-1.2.1-1.4, the shared process model of the current airspace state is now an additional contributing factor that could lead a pilot to disregard the onboard sensors. In addition to another controller (e.g. pilot in another aircraft) providing input, the shared process model essentially serves as another input. Based on either of those inputs, the Piloting Controller could decide to disregard the onboard sensors because they assumed that those inputs are more accurate than the sensor data being received. This scenario therefore needs to be updated because the shared process model between EMS Operations and ATC and the Piloting Controller could lead the Piloting Controller to wrongly ignore real-time sensor feedback.

These updates to the loss scenarios therefore demonstrate how sharing the process model of the current airspace state between EMS Operations and ATC and the Piloting Controller makes these scenarios more difficult to prevent because sharing the current airspace state process model introduces additional ways in which the Piloting Controller could receive conflicting feedback or be led to wrongly ignore feedback. As a result, it will be even more important to ensure that sufficient constraints are placed on the Piloting Controller's behavior to prevent these updated loss scenarios from occurring.

In addition to updating existing scenarios to account for additional causal factors introduced by this architecture option, a control action was also added from EMS Operations and ATC to the Piloting Controller as shown in the control structure (Figure 21). This control action was added because EMS Operations and ATC is sharing situational information with the Piloting Controller to help it keep its process model of the current and future airspace state updated. It is therefore necessary to also perform an STPA analysis on this new control action to determine how the addition of this control action could result in unsafe system behavior. Although a full analysis will not be performed in this case study, some example UCAs and scenarios are shown below to illustrate how the STPA analysis might be done.

Table 22 shows some example UCAs that are associated with the "Updated Situational Information" control action.

Table 22: UCAs for Updated Situational Information Control Action

Control Action	Providing	Not Providing	Provide too early/too late	Applied too long/Stopped too soon
Updated Situational Information	UCA-2-1: EMS Operations and ATC provides updated situational information when the situational information does not accurately represent the state of the airspace [H-1, H-3, H-5] UCA-2-2: EMS Operations and ATC provides updated situational information that is difficult to read/interpret [H-1, H-3, H-5]	UCA-2-5: EMS Operations and ATC does not provide updated situational information when such information is not available to the PC from an alternative source	UCA-2-6: EMS Operations and ATC provides updated situational information too early before an impending change is made/occurs UCA-2-7: EMS Operations and ATC provides updated situational information too late after an	N/A

	UCA-2-3: EMS Operations and ATC provides updated situational information that is either not self-consistent or conflicts with other data sources UCA-2-4: EMS Operations and ATC provides updated situational information for the wrong part of the airspace		actuator movement has been selected based on incorrect situational information	

Loss scenarios can then be generated for each of the UCAs identified. Some example loss scenarios are as follows:

UCA-2-1: EMS Operations and ATC provide updated situational information when the situational information does not accurately represent the state of the airspace

CS-2-1-1: EMS Operations and ATC provide updated situational information despite receiving feedback that indicates that the situational information does not accurately represent the state of the airspace. This might occur if:

- Although feedback was received that indicates that the situational information is inaccurate, the inaccurate version is provided before the process model can be updated with the new information
- Although EMS Operations and ATC recognized that the situational information does not accurately represent the state of the airspace, EMS Operations and ATC decide that either it is accurate enough to use or that the inaccurate information is better than no information and provides that situational information as an update anyway.
- When EMS Operations and ATC receive situational information updates (e.g. from previous missions, from reconnaissance aircraft), if it has no way to evaluate the accuracy of that information, the EMS Operations and ATC might assume that the information is accurate and not recognize that the situational information it is receiving contains inaccuracies even though the inaccuracies are detectable in the data

CS-2-1-2: EMS Operations and ATC receive feedback that does not indicate that the updated situational information does not accurately represent the state of the airspace. This might occur if:

- When inaccuracies are observed by users of the data, those inaccuracies are not reported back to the EMS Operations and ATC (e.g. due to time constraints if they are observed during a flight)
- The EMS Operations and ATC might not have the resources/capability to perform an independent assessment of the situational information it receives and is therefore either unable to evaluate the accuracy or is forced to assume a level of accuracy

CS-2-1-3: EMS Operations and ATC do not provide updated situational information but updated situational information is received by the Piloting Controller. This might occur if:

- Although the EMS Operations and ATC correctly recognize that the updated situational information does not accurately represent the state of the airspace, another aircraft or controller that also has access to that information does not recognize that and provides the Piloting Controller with that information. As a result, the Piloting Controller is still provided with that situational information even though the EMS Operations and ATC do not provide it

The above results therefore show that even in this short example analysis, the addition of the control action to provide updates to the situational information from EMS Operations and ATC to the Piloting Controller introduces 7 new UCAs and, for just the first UCA, 5 new loss scenarios. If this analysis were completed, the actual number of new UCAs and scenarios introduced by the new control action would be even higher. This analysis along with the updates to the existing scenarios therefore shows that the sharing of the process model parts and the addition of the new control action offers the potential to reduce the workload of the Piloting Controller and improve the situational information available to it. However, those benefits come at the cost of additional ways that the system could behave in an unsafe way and more requirements that must be added to adequately constrain the system behavior to ensure safety.

4.5.6 Comparison of Architecture Options

The two architecture options can now be compared to assess the tradeoffs between architecture options that would inform a decision on which architecture to select for further system development. By comparing the assignments of the process model parts and the responsibilities, two tradeoffs can be observed.

Comparing Assignment of Process Model Parts

The first tradeoff between the architecture options is that although the shared process model parts might provide the Piloting Controller with access to better process model information, that information may also serve as an additional source of conflict or wrong decision making that is avoided in option 1. As shown in Table 23, the main differences in the assignment of process model parts for the two architecture options are that the Current Airspace State and Anticipated Future Airspace State process model parts are only assigned to the Piloting Controller in option 1 but are shared between the Piloting Controller and EMS Operations and ATC in option 2.

Table 23: Comparison of Process Model Part Assignments for Part 1 Architectures

Process Model	Option 1 Assigned to				Option 2 Assigned to			
	EMSATC	MP	PC	AS	EMSATC	MP	PC	AS
Current Aircraft Mission Readiness		X	X			X	X	
Current Aircraft System State			X	X			X	X
Current Aircraft Navigation State	X		X		X		X	
Expected Safe Departure State	X	X	X		X	X	X	
Current Weather Conditions			X				X	
Current Airspace State			X		X		X	
Anticipated Future Airspace State			X		X		X	
Model of System Behavior	X		X	X	X		X	X

EMSATC – EMS Operations and ATC | MP – Maintenance
PC – Piloting Controller | AS – Aircraft Subsystems

In option 1, since the Piloting Controller does not share the highlighted process model parts with other controllers, there are fewer opportunities for the information in the process model to lead to unsafe behavior compared to option 2. However, those process model parts only contain information that the Piloting Controller has direct access to during flight. By contrast, in option 2, since EMS Operations and ATC shares the two process model parts, EMS Operations and ATC can update those process model parts with information that the Piloting Controller would otherwise not have access to. As a result of those process model parts being shared, the Piloting Controller could then gain access to that information. The Piloting Controller therefore has access to better information in option 2. However, the tradeoff is that the additional information can also become a source of conflict or unsafe decision making as described in the analysis of architecture option 2.

Comparison of Responsibility Assignments

The second tradeoff between the architecture options is that although option 2 has the potential to reduce the workload of the Piloting Controller, sharing responsibility assignments between the Piloting Controller and EMS Operations and ATC introduces the potential for unsafe behavior by

EMS Operations and ATC to cause the Piloting Controller itself to make unsafe decisions. This can be observed by comparing the assignment of SR-18 in the two architectures as shown in Table 24 as well as by comparing the assignment of SR-1 and SR-9 for the current airspace state and anticipated future airspace state process model parts as shown in Table 25.

Table 24: Comparison of SR-1 for Part 1 Architectures

Resp ID	Responsibility	Option 1 Assigned to				Option 2 Assigned to			
		EMSATC	MP	PC	AS	EMSATC	MP	PC	AS
SR-18	Detect all objects and other aircraft in the environment under all DVE conditions at all times [SLR-20]			X		X		X	

Table 25: Comparison of SR-9 and SR-18 for Part 1 Architectures

Resp ID	Responsibility	Option 1 Process Model Parts		Option 2 Process Model Parts	
		Current Airspace State	Anticipated Future Airspace State	Current Airspace State	Anticipated Future Airspace State
SR-1	Determine if feedback received about the aircraft's mission readiness, state of the aircraft and the airspace is too old [SLR-1]	PC	PC	PC EMSATC	PC EMSATC
SR-9	Ensure that all data needed to determine the state of the aircraft, state of the airspace and the environmental conditions around the aircraft under all DVE conditions is available at all times [SLR-9, SLR-15]	PC	PC	PC EMSATC	PC EMSATC

As shown in Table 24 and Table 25, the differences between the responsibility assignments in options 1 and 2 are primarily that the responsibilities for detecting other aircraft and objects in the airspace and updating the process model of the current and future airspace state are shared with EMS Operations and ATC in option 2 whereas the Piloting Controller alone performs those responsibilities in option 1. As discussed in the analysis of option 1, flying the aircraft in DVE imposes additional workload on the Piloting Controller because some aspects of maintaining safe and stable flight in DVE conditions are more difficult to carry out or new tasks must be performed. By contrast, in option 2, sharing these three responsibilities between EMS Operations and ATC and Piloting Controller. This has the potential to reduce the workload on the Piloting Controller because the Piloting Controller is no longer solely responsible for carrying out these responsibilities. In addition, sharing these responsibilities may also help to reduce the risk of stringent requirements because EMS Operations and ATC may be able to help to detect some

objects that the Piloting Controller and Aircraft Subsystems might not be able to. For example, EMS Operations and ATC may be able to determine the location of immobile objects such as power lines and cell towers more easily than the Piloting Controller and Aircraft Subsystems can using real-time sensing and detection algorithms

However, this benefit comes at a cost. As discussed in the analysis of option 2, this sharing of responsibilities introduces an additional interaction between EMS Operations and ATC and the Piloting Controller that is not present in option 1. As a result of this additional interaction, unsafe behavior by EMS Operations and ATC can potentially influence the behavior of the Piloting Controller, leading the Piloting Controller to make unsafe decisions that it might not have made on its own. For this reason, new UCAs and scenarios are introduced in option 2 compared to option 1 that could be avoided if these responsibilities were not shared. This therefore shows that although sharing responsibilities has the potential to reduce the workload of the Piloting Controller, the cost of gaining this benefit is that more requirements will need to be added to enforce additional constraints to prevent unsafe system behavior.

Considering these tradeoffs together, Table 26 summarizes the benefits and costs associated with each architecture option.

Table 26: Summary of Benefits and Costs for Part 1 Architecture Options

	Benefits	Costs
Option 1	Feedback is primarily integrated by the Piloting Controller, reducing the opportunities for conflicting process model information Simpler architecture with less sharing of process model parts and responsibilities	High workload imposed on the Piloting Controller Stringent performance requirements imposed on Piloting Controller and feedback mechanisms
Option 2	Better information about the current and anticipated future airspace state Potentially reduced workload for the Piloting Controller Reduced reliance on the Piloting Controller to perform all responsibilities necessary to ensure safe flight in DVE conditions	Introduces additional sources of conflicting inputs that can lead to wrong or unsafe decisions Unsafe behavior of EMS Operations and ATC can lead to unsafe behavior by the Piloting Controller

Based on the tradeoffs discussed above and any other inputs that might be needed for decision making (e.g. cost of each architecture option, budget available, level of technical risk), a system designer can then make an informed architectural decision about which architecture option should be chosen. Eventually, a decision will need to be made to proceed with either option 1 or option 2 for further system development.

For the purposes of this case study, it will be assumed that option 2 is chosen to proceed with Part 2 of the architecture creation process. Option 2 is chosen for the following reasons:

- The workload and capability requirements imposed on the Piloting Controller is too great in option 1
- The benefits of sharing some of the responsibilities and process model parts are worth the costs. This avoids being so heavily reliant on the Piloting Controller to ensure safe flight and provides the Piloting Controller with better information

4.6 Step 4, Part 2: Creating and Assessing Architecture Options for the Piloting Controller

With option 2 from Part 1 having been selected as the architecture for the FOS, in Part 2 of the architecture creation process, the responsibilities and process model parts that were assigned to the Piloting Controller must now be divided up between the Human Pilot and Automated Software-Enabled Controller (ASEC) that comprise the Piloting Controller. Except for SR-7, SR-10, SR-11 and SR-12, the remaining twenty responsibilities and all the process model parts that were identified as part of the system-level behavior are assigned to or shared with the Piloting Controller. Therefore, these twenty responsibilities will now be assigned to the human pilot or the ASEC to create three further architecture options. In this part of the architecture creation process, because the responsibilities and process model part are assigned to a human or automated controller, more detailed analyses of each architecture option can be conducted that more clearly incorporate considerations of human factors and other considerations into the analysis.

4.6.1 Overview of Architecture Options

The three architecture options for how the responsibilities and process model parts could be assigned are as follows:

1. Option 1: Human-Piloted Aircraft with ASEC as a Decision Aid
2. Option 2: ASEC Proposes Flight Trajectories for Human Pilot to Execute
3. Option 3: Human Pilot Supervises Automated ASEC's Control of Flight

As implied by the names of the architecture options, they represent architecture options that use an increasing level of automation to assist the human pilot in safely flying the aircraft in DVE conditions to execute a mission. Option 1 is the architecture that relies the least on automation where the aircraft is still flown primarily by a human pilot and the ASEC serves as a decision aid to help the human pilot make decisions about flight path and trajectory by providing feedback such as detected objects and other aircraft in the airspace or prevailing weather conditions.

By contrast, option 3 is the architecture that relies the most on automation to assist the pilot in flying the aircraft. In this architecture option, the ASEC is assigned many of the core piloting responsibilities to relieve the pilot of the more mundane and routine aspects of flight. The human pilot therefore serves more of a supervisory role and can monitor the decisions made by the ASEC and assist the ASEC in more complex decision making and problem solving tasks.

Option 2, then, is the architecture that uses a moderate level of automation to assist the pilot in flying the aircraft. Unlike option 1, the ASEC is more involved in selecting appropriate flight trajectories based on mission requirements, airspace hazards and prevailing weather conditions. However, unlike option 3, the human pilot is still an operator of the system because they are in direct control of the aircraft.

4.6.2 Architecture Option 1: Human-Piloted Aircraft with ASEC as a Decision Aid

As described in the overview, this architecture option relies the least on automation and the human pilot performs most of the tasks necessary to fly the aircraft safely while the ASEC serves as a decision aid to help them in selecting an appropriate trajectory and flight path. As a result, most of the process model parts and responsibilities are assigned to the human pilot with some responsibilities also shared with the ASEC.

Assignment of Process Model Parts

Since all eight process model parts identified in the system-level behavior information were assigned to the Piloting Controller in part 1 of the architecture creation process, all eight process model parts must now be assigned to the human pilot or ASEC. These assignments are shown in Table 27 along with the underlying rationale and assumptions for each assignment.

Table 27: Sample Assignments of Process Model Parts for Part 2 Architecture Option 1

Process Model	Assigned to		Rationale/Assumptions
	Pilot	ASEC	
Current Aircraft Mission Readiness	X		Much of this information is not provided in a way that is easily interpreted by automation. A human would be better positioned to interpret this data quickly
Current Aircraft System State	X	X	Much of this information is generated automatically and can be easily parsed by automation. However, the pilot is flying the aircraft and therefore must maintain this part as well.
Current Aircraft Navigation State	X	X	
Expected Safe Departure State	X		Assumes that most of this information is more easily parsed by a human rather than translated for automation
Current Weather Conditions	X	X	Much of this information is generated from sensors and can be easily parsed by automation. However, the pilot is flying the aircraft and therefore must maintain this part as well.
Current Airspace State	X	X	Sensor data needed to determine this is more easily parsed by automation. However, because this data must be integrated with intent information and coordinated with other aircraft, the human pilot must maintain this model as well. In addition, the human pilot needs this information to fly the aircraft
Anticipated Future Airspace State	X		The complexity of intent information requires a human to parse or converse with other controllers and would be difficult to automate
Model of System Behavior	X		The parameters in this part of the process model mainly pertain to data interpretation that would be performed by the human pilot and therefore only the human pilot needs this process model part

All the process model parts are assigned to the human pilot because they are flying the aircraft and will need to use all the process model parts. In addition, since the ASEC serves as a decision aid, it shares four of the process model parts: Current Aircraft System State, Current Aircraft Navigation State, Current Weather Conditions and Current Airspace State. These four process model parts are shared because the ASEC receives data from the aircraft subsystems on prevailing weather conditions, detections of other objects and aircraft in the airspace, the position of the aircraft and the state of the subsystems to display to the pilot. Thus, the ASEC also maintains a copy of these parts of the process model in addition to the human pilot.

Assignment of Non-Feedback-Validation Responsibilities

In this architecture option, since the human pilot is primarily flying the aircraft with the ASEC serving only as a decision aid, most of the non-feedback-validation responsibilities are assigned to the pilot because they are responsibilities that are more easily performed by the human pilot than the ASEC that is only minimally involved in the task of flying the aircraft. Examples of the responsibilities that are assigned only to the pilot are shown in Table 28.

Table 28: Non-Feedback-Validation Responsibilities Assigned to the Pilot for Part 2 Architecture Option 1

Resp. ID	Responsibility	Assigned to		Rationale/Assumptions
		Pilot	ASEC	
SR-4	Confirm that all aspects of the aircraft state match the expected safe departure state before providing departing [SLR-4]	X		Expected safe departure state is most easily obtained by human pilot instead of ASEC
SR-16	Account for the current and future movements of other aircraft in the vicinity when selecting actuator movements [SLR-17]	X		Coordinating with other aircraft and adapting to work with their movements is more easily done in real-time by a human than automation
SR-17	Ensure that aircraft movements are selected such that sufficient reaction time is available if intent or movements of other aircraft are not what was expected, even if no violation of minimum separation is initially expected [SLR-18]	X		Since this involves integrating intent information alongside current airspace state and aircraft state information to decide the best path to take and the best actuator movements to apply, this is best done by a human pilot
SR-20	Select actuator movements that minimize the risk of violation of minimum separation when information about the state of the airspace is in a degraded condition (e.g. delayed) [SLR-21]	X		Similar to SR-17, this decision making is complex enough and the definition of risk minimization is likely vague enough that it is better performed by a human pilot than by automation
SR-23	Ensure that a viable flight path is always available to avoid any violations of minimum separation [SLR-24]	X		Assumes that automation sophistication will not be sufficient to select flight paths under all conditions and therefore it makes more sense for a human pilot to retain this function
SR-24	Select a viable flight path that avoids all possible violations of minimum separation [SLR-26]	X		Assumes that automation sophistication will not be sufficient to select flight paths under all conditions and therefore it makes more sense for a human pilot to retain this function

As can be seen in the rationale provided in Table 28, these responsibilities are assigned only to the human pilot because they either involve complex or ambiguous decision making that is best performed by a human pilot or they involve the use of feedback or information that is more easily interpreted by a human pilot rather than the ASEC. For example, as described in the decision-making strategy for SR-16, SR-17, SR-23 and SR-24, these responsibilities involve a complex and sometimes ambiguously defined decision based on consideration of multiple factors to select appropriate actuator movements that do not cause a violation of minimum separation. As such, the adaptive and flexible nature of human cognition may be more suited to making these decisions than the more rigid decision-making processes of the ASEC. Similarly, for SR-4 and SR-17, these responsibilities require the use of process model parts and feedback that are more easily interpreted by a human pilot than by the ASEC. As such, it would be easier for a human pilot to make use of the feedback to carry out these responsibilities instead of the ASEC.

By contrast, some responsibilities are either shared with the ASEC or only assigned to the ASEC and examples of these assignments are shown in Table 29.

Table 29: Other Non-Feedback-Validation Responsibilities for Part 2 Architecture Option 1

Resp. ID	Responsibility	Assigned to		Rationale/Assumptions
		Pilot	ASEC	
SR-15	Determine if the state of the airspace changes between the time that it was checked and the commencement of a maneuver [SLR-16]	X	X	Since this is relatively simple checking of the maneuver path, the ASEC is able to warn the human pilot of potential collisions but the human pilot is expected to perform an independent verification as well
SR-19	Detect slow or subtle changes in the state of the aircraft under all DVE conditions at all times [SLR-20]		X	Given the potential magnitude of these changes in the state of the aircraft, it may be difficult to design appropriate feedback mechanisms with sufficient saliency for a human pilot to notice those changes and they would be easier detected by automation
SR-21	Respond quickly and with appropriate magnitude to disturbances to prevent unintended movement of the aircraft [SLR-22]	X	X	Assumes that the ASEC will not be able to stabilize the aircraft sufficiently under all DVE conditions and that a human pilot would be needed in some situations Also assumes that ASEC will be able to help stabilize the aircraft under limited DVE conditions and therefore the ASEC will sometimes be used
SR-22	Respond quickly enough and with appropriate magnitude to select and effect the desired flight path under the given environmental conditions [SLR-23, SLR-25]	X	X	

As might be expected, since the ASEC is simply a decision aid for the human pilot, only a few of the non-feedback-validation responsibilities are assigned to it. In some cases, the responsibilities are shared so that the ASEC can assist the human pilot by helping to warn them (SR-15) or helping them to control the aircraft under easier situations (SR-21 and SR-22). In other cases such as SR-19, the ASEC is better suited to monitor for conditions that would be difficult for a human to monitor such as slow or subtle changes in the state of the aircraft.

Assignment of Feedback Validation Responsibilities

Finally, the feedback validation responsibilities need to be assigned to the human pilot and the ASEC. Like Part 1, whenever a particular responsibility is not expected to be applicable for a particular process model part, "N/A" is used to indicate that no assignment is made.

For this architecture option, because the ASEC serves as a decision aid for the human pilot, many of the responsibilities are shared between the ASEC and human pilot such that the ASEC and human pilot carry out a given responsibility for different process model parts. In this way, the ASEC serves as a decision aid because it carries out some of these responsibilities for some of these process model parts instead of requiring the human pilot to carry out all the responsibilities for every process model part. Some examples of feedback validation responsibility assignments are shown in Table 30.

Table 30: Assignment of Feedback Validation Responsibilities for Part 2 Architecture Option 1

<table>
<tr><th>Resp ID</th><th>Responsibility</th><th>Assignments for Each Process Model Part</th><th>Rationale/Assumptions</th></tr>
<tr><td>SR-3</td><td>Validate the inputs or feedback received from other controllers before using that input or feedback to update process models [SLR-3]</td><td>
<table>
<tr><td>Aircraft Mission Readiness</td><td>Pilot</td></tr>
<tr><td>Current Aircraft System State</td><td>ASEC</td></tr>
<tr><td>Current Aircraft Navigation State</td><td>Pilot</td></tr>
<tr><td>Expected Safe Departure State</td><td>N/A</td></tr>
<tr><td>Current Weather Conditions</td><td>Pilot</td></tr>
<tr><td>Current Airspace State</td><td>Pilot</td></tr>
<tr><td>Anticipated Future Airspace State</td><td>Pilot</td></tr>
<tr><td>Model of System Behavior</td><td>N/A</td></tr>
</table>
</td><td>The variety of possible inputs made by other aircraft or controllers would be more easily handled by a human, except for the aircraft system state which would be more easily handled by automation since the system state is already generated and monitored by automation

Expected safe departure state and model of system behavior are marked as N/A because they were not assigned to the Piloting Controller in part 1</td></tr>
</table>

<table>
<tr><th>Resp ID</th><th>Responsibility</th><th>Assignments for Each Process Model Part</th><th>Rationale/Assumptions</th></tr>
<tr><td>SR-9</td><td>Ensure that all data needed to determine the state of the aircraft, state of the airspace and the environmental conditions around the aircraft under all DVE conditions is available at all times [SLR-9, SLR-15]</td><td><table>
<tr><td>Aircraft Mission Readiness</td><td>Pilot</td></tr>
<tr><td>Current Aircraft System State</td><td>ASEC</td></tr>
<tr><td>Current Aircraft Navigation State</td><td>ASEC</td></tr>
<tr><td>Expected Safe Departure State</td><td>N/A</td></tr>
<tr><td>Current Weather Conditions</td><td>ASEC</td></tr>
<tr><td>Current Airspace State</td><td>ASEC</td></tr>
<tr><td>Anticipated Future Airspace State</td><td>Pilot</td></tr>
<tr><td>Model of System Behavior</td><td>N/A</td></tr>
</table></td><td>Monitoring the inputs for aircraft system state, navigation state and environmental conditions is a repetitive task best done by automation.

For anticipated future airspace state and aircraft mission readiness, since that feedback may include verbal radio communication, the pilot shares the responsibility to ensure that needed information is available.</td></tr>
<tr><td>SR-13</td><td>Distinguish useful feedback from noise in feedback data [SLR-13]</td><td><table>
<tr><td>Aircraft Mission Readiness</td><td>N/A</td></tr>
<tr><td>Current Aircraft System State</td><td>ASEC</td></tr>
<tr><td>Current Aircraft Navigation State</td><td>ASEC</td></tr>
<tr><td>Expected Safe Departure State</td><td>N/A</td></tr>
<tr><td>Current Weather Conditions</td><td>ASEC</td></tr>
<tr><td>Current Airspace State</td><td>Pilot
ASEC</td></tr>
<tr><td>Anticipated Future Airspace State</td><td>Pilot</td></tr>
<tr><td>Model of System Behavior</td><td>N/A</td></tr>
</table></td><td>Especially when identifying weak signals, automation is better at the pattern recognition necessary to extract weak signals

Assumes that the risk of false negative detections of weak signals is acceptably low

Mission readiness, expected safe departure state and model of system behavior do not rely on real-time feedback and therefore are not affected by noise</td></tr>
</table>

<table>
<tr><th>Resp ID</th><th>Responsibility</th><th>Assignments for Each Process Model Part</th><th>Rationale/Assumptions</th></tr>
<tr><td>SR-14</td><td>Process sensor data and use it to update its process model sufficiently quickly [SLR-14]</td><td>
<table>
<tr><td>Aircraft Mission Readiness</td><td>Pilot</td></tr>
<tr><td>Current Aircraft System State</td><td>ASEC</td></tr>
<tr><td>Current Aircraft Navigation State</td><td>Pilot
ASEC</td></tr>
<tr><td>Expected Safe Departure State</td><td>N/A</td></tr>
<tr><td>Current Weather Conditions</td><td>Pilot
ASEC</td></tr>
<tr><td>Current Airspace State</td><td>Pilot
ASEC</td></tr>
<tr><td>Anticipated Future Airspace State</td><td>Pilot</td></tr>
<tr><td>Model of System Behavior</td><td>N/A</td></tr>
</table>
</td><td>For everything but the aircraft system state, the pilot is primarily the one updating the process model parts and the ASEC only assists in the aircraft navigation state, weather conditions and airspace state.

Expected safe departure state and model of system behavior are marked as N/A because they were not assigned to the Piloting Controller in part 1</td></tr>
</table>

Architecture Option 1 Control Structure

The resulting control structure for this architecture option is shown in Figure 22.

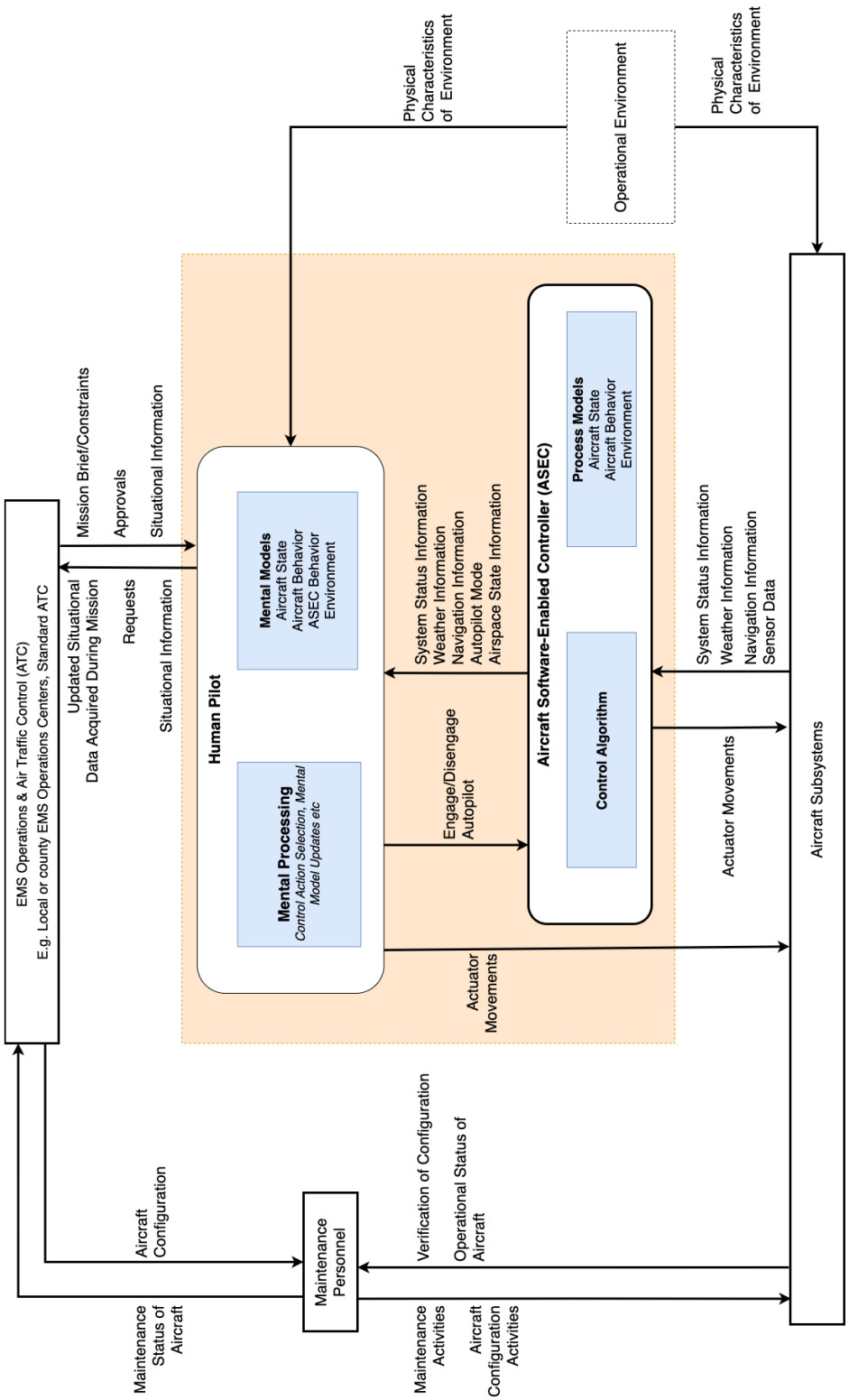


Figure 22: Part 2 Architecture Option 1 Control Structure

4.6.3 Architecture Analysis of Option 1

Like option 1 in Part 1 of the architecture creation process, this option makes no changes to how human pilots and aircraft automation divide up the responsibilities needed to fly aircraft today. In essence, this architecture option represents the architecture where human pilots fly aircraft in DVE conditions in much the same way as they do today under non-DVE conditions. The only difference is that the human pilots have access to more data provided by sensor suites onboard the aircraft than is available in today to help detect objects and other aircraft in the airspace instead of being reliant only on direct visual feedback and transponder or radio-based communication. As a result, the STPA analysis of this architecture option will be similar to the analysis presented in Section 3, but with additions or updates to the loss scenarios based on additional human factors and other considerations that can be applied now that responsibilities are assigned to the human pilot or the ASEC.

There are two challenging aspects of this architecture that can be observed. The first is that because many of the responsibilities are assigned to the human pilot, the feedback mechanisms that the human pilot uses to obtain information needed for decision making must be designed to avoid known decision-making biases that can affect the pilot's ability to make correct decisions using that feedback. To help identify the decision making biases and heuristics that need to be avoided to prevent unsafe behavior, the loss scenarios in the STPA analysis should be updated to account for human factors reasons that a human pilot might issue unsafe control actions. Table 31 shows two responsibilities that are assigned to the human pilot that describe constraints on the use of feedback as well as the associated loss scenarios.

Table 31: Example Constraints and Scenarios Related to the Use of Feedback

Resp. ID	Responsibility	Associated Scenario ID	Associated Scenario
SR-2	Account for prevailing DVE conditions and their possible effect on feedback sources when making use of feedback information [SLR-2]	CS-1.1.1-1.1.2	The piloting controller believes it is receiving conflicting feedback because DVE conditions have degraded the accuracy of or obscured some (but not all) feedback sources. The piloting controller therefore chooses to ignore the feedback showing the presence of the other aircraft or object, believing that it is a false positive or erroneous. The piloting controller is especially susceptible to this if the majority of sources appear to agree that no aircraft is present and fewer sources appear to show otherwise. As a result, the piloting controller updates its process model based only on what it wrongly believes to be the correct feedback. [SLR-2]

Resp. ID	Responsibility	Associated Scenario ID	Associated Scenario
SR-3	Validate the inputs or feedback received from other controllers before using that input or feedback to update process models [SLR-3]	CS-1.1.1-1.1.3	Alternatively, if the piloting controller receives communication from another controller (e.g. another aircraft, ground personnel, EMS Operations and ATC) indicating that the aircraft is in a safe departure state, the piloting controller may assume that the communication from the other controller must be accurate. This may even be used as proof that the feedback showing that the aircraft is not in a safe departure state should be ignored as out-of-date or incorrect. As a result, the piloting controller uses the communication from the other controller to update its process model and therefore wrongly believes that the aircraft is in a safe departure state. [SLR-3]

Since SR-2 and SR-3 are assigned to the human pilot, their associated loss scenarios can be updated to include human factors considerations for how these loss scenarios could occur. For example, the human pilot might wrongly believe they are receiving conflicting feedback (scenario CS-1.1.1-1.1.2) if a pilot is initially presented with detection information from the ASEC showing that no aircraft is present in the nearby airspace because DVE conditions have obscured the ability to detect the other aircraft. As a result, a known human decision-making bias called cue primacy [30] can cause the pilot to be more inclined to continue to believe that no aircraft is present even if those same sensors detect a nearby aircraft several seconds later. In addition, another human decision-making bias known as inattention to later cues [30] can then cause the pilot to ignore subsequent feedback that does detect the nearby aircraft. Alternatively, if a pilot has experienced unreliable false positive detections from the sensors in the past, they may use that experience to rationalize ignoring feedback in a new situation, wrongly believing that they are experiencing a situation like the one they experienced in the past where detections turned out to be false positives. These updates to scenario CS-1.1.1-1.1.2 therefore show that if the performance of the sensor suite is inadequate, human decision-making biases may lead the human pilot to make decisions that can lead to unsafe control actions being provided.

Similarly, scenario CS-1.1.1-1.1.3 can be updated to include the impact of the process model of the state of the airspace that is shared with EMS Operations and ATC. Since the shared process model can be used by the human pilot as another input available to them, confirmation bias can cause the pilot to accept the input from the shared process model if it is aligned with their initial belief that no aircraft or object was detected in the airspace. As a result, the pilot may ignore more direct, seemingly contradictory feedback from the aircraft sensors showing the presence of that aircraft or object. Alternatively, the pilot's past experiences may also lead them to believe that the shared database is always more accurate and place an inappropriate level of trust in the shared database even when that database is wrong. In either case, the pilot therefore chooses to ignore correct feedback from the aircraft sensors even when that feedback reflects the correct

state of the airspace. These updates to CS-1.1.1-1.1.3 therefore show that the feedback mechanisms (both the sensor suite and the shared database) used by pilots must be designed to ensure that pilots place an appropriate level of trust in the feedback they receive and do not simply default to always trusting or being mistrustful of one feedback source over another.

These updates to the two loss scenarios thus show that, by assigning responsibilities like SR-2 and SR-3 to the human pilot, the performance (e.g. reliability, accuracy) of the sensor suite is extremely important because the performance of the sensor suite can impact the pilot's ability to perceive the feedback they need, make correct decisions about when they are receiving conflicting feedback and decide how to use the feedback to update their mental model. If the pilot's ability to correctly integrate relevant feedback to update their mental model is impaired by decision-making biases, misplaced levels of trust or high stress and workload situations, the human pilot may not be as effective in enforcing the necessary safety constraints to safely fly the aircraft in DVEs. These loss scenarios therefore serve as the rationale for identifying the necessary performance requirements for the sensor suite during detailed system design.

The other challenging aspect of this architecture option is that this architecture option imposes high workloads on the human pilot to carry out all the responsibilities assigned to them and the STPA analysis and system-level behavior information highlight three aspects that contribute to the high workload. The first two aspects are illustrated in Table 32.

Table 32: Responsibilities and scenarios associated with pilot's maintenance of their mental model of the airspace

Resp. ID	Responsibility	Associated Scenario ID	Associated Scenario
SR-14	Process sensor data and use it to update its process model sufficiently quickly [SLR-14]	CS-1.2.1-1.3	DVE causes delays in the piloting controller interpreting the sensor data and using the sensor data to update its process model. As a result, the piloting controller has the wrong process model of the airspace around the aircraft until it is able to update its process model. [SLR-14]
SR-22	Respond quickly enough and with appropriate magnitude to select and effect the desired change in flight path under the given environmental conditions [SLR-23, SLR-25]	CS-1.5.1-1.1	The piloting controller is unable to select new actuator movements quickly enough to avoid violation of minimum separation. This might occur if the piloting controller recognizes the imminent violation too late or takes too long to select new actuator movements [SLR-23]
SR-24	Select a viable flight path that avoids all possible violations of minimum separation [SLR-26]	CS-1.5.4-2	The piloting controller selects actuator movements that avoid one violation of minimum separation but causes another one [SLR-25, SLR-26]

The first aspect that increases the workload for the human pilot is the increased cognitive effort required by the human pilot to validate feedback and update their mental model. One example of such a requirement is SR-14 shown in the first row of Table 32. For the human pilot to carry out SR-14 and avoid being delayed in interpreting sensor data as described in CS-1.2.1-1.3, the human pilot will likely need to expend significant mental resources to process sensor data and use it to update its process model at a sufficiently fast rate as described in the system-level behavior. In addition, the cognitive effort may be even higher if the human pilot has difficulty making use of the sensor data. If, for example, the pilot must manually integrate detections from multiple sensor modalities to develop an overall understanding of what the sensors are detecting in the airspace, the human pilot will need to spend additional mental effort to perceive and deconflict the various pieces of feedback. This could delay their ability to update their mental model of the state of the airspace in a timely manner. As such, the cognitive effort needed to perform these types of responsibilities can result in a high workload imposed on the human pilot.

The second aspect that increases the workload for the human pilot is the potentially difficult conditions under which the human pilot is expected to make decisions. One category of difficult decisions is those that must be made quickly and accurately with limited time to consider alternatives and SR-22 shown in the second row of Table 32 is one example of this type of responsibility. To carry out SR-22, the human pilot would need to be able to consider all of the various factors (e.g. weather conditions, location of objects and aircraft in the airspace) and select the correct magnitude and direction of actuator movements within a short period of time as described in the system-level behavior information. If the pilot has extensive flight experience in DVE conditions, they may be able to make use of skill- or rule-based decision-making [30] to select appropriate actuator movements quickly based on the feedback they are perceiving. However, if the pilot has little experience flying the aircraft in DVE or an experienced pilot performs a flight under novel or rarely experienced conditions, they may have to use knowledge-based decision making [30] and make use of their knowledge and past experience along with techniques such as mental simulation to select appropriate actuator movements. Under these challenging decision-making conditions, if a pilot takes too long or selects the wrong magnitude of actuator movement (CS-1.5.1-1.1), an accident or loss could occur. As such, by assigning responsibilities like SR-22 to a human pilot, the conditions under which the human pilot must make decisions can require a high workload to carry out those responsibilities.

Another category of difficult decisions are complex decisions that must be made under uncertainty or incomplete information and SR-24 shown in Table 32 is one example of this type of responsibility. To carry out SR-24 and avoid CS-1.5.4-2, the pilot must select actuator movements without certain knowledge of how the other aircraft will move while ensuring that all possible violations of minimum separation are avoided. In these scenarios, especially if DVE conditions prevent the pilot from directly observing other aircraft or objects in the airspace, selecting appropriate actuator movements can be especially challenging. Not only is the pilot reliant on feedback from sensors and other controllers to maintain enough situational awareness to make these decisions but they also must consider a variety of possible future states of the airspace to select one that they believe will avoid all possible violations of minimum separation. The pilot therefore must make the best use of the available information to select appropriate

actuator movements that will minimize the likelihood that they encounter another aircraft or object that was not expected or detected. As such, although humans are especially well-suited for making decisions under uncertainty using vaguely defined criteria, the limited time or information available can make these decisions cognitively challenging, increasing the workload imposed on the human pilot when they are assigned these responsibilities.

Finally, the third aspect that increases the workload for the human pilot is the larger mental model that they must maintain to operate the aircraft safely. As described in Section 4.4.3 and shown in full in Table 50 of Appendix D, the full process model identified for this case study consists of eight main parts and numerous sub-parts, all of which must be maintained and updated continuously throughout the flight. Although some parts of this mental model exist regardless of the existence of DVE conditions, some parts of this mental model are added because of the additional responsibilities needed to fly aircraft in DVE. Table 33 shows some examples of such parts of the mental model and the reason they are included.

Table 33: Examples of Mental Model Parts Included Due to the Presence of DVE Conditions

<table>
<tr><th>Mental Model Part</th><th>Rationale for Inclusion (From System-Level Behavior)</th></tr>
<tr><td>Known effects of DVE on feedback sources</td><td>This is needed by SR-2 to determine when feedback sources are no longer reliable because of DVE conditions. If DVE conditions were not present, this would not be needed because it would not be necessary for pilots to assess the effects of DVE on feedback sources in real time</td></tr>
<tr><td>Typical noise expected for each data source under DVE conditions</td><td rowspan="2">These are needed by SR-13 to help in distinguishing useful feedback from noise under DVE conditions that might increase the amount of noise. If DVE conditions were not present, this part of the mental model may not be needed.</td></tr>
<tr><td>Expected signal strength for each data source under DVE conditions</td></tr>
<tr><td>Actuator movements selection guidelines for handling the aircraft in DVE conditions</td><td>This is needed by SR-21 and SR-22 to enable the selection of appropriate actuator movements under DVE conditions. If DVE conditions were not present, only normal aircraft handling procedures would be needed</td></tr>
</table>

As a result of these additional parts of the process model that are added to enable flight in DVEs, the mental model that is maintained and updated by the human pilot is larger than the one they would have had to maintain and update if DVE conditions were not present, contributing to the increased workload imposed on the human pilot.

In summary, the analysis of this architecture option highlights some of the challenges of implementing this architecture option. Not only must the sensor and feedback mechanisms be designed to avoid known biases in human decision-making but the system must also be designed to ensure that the increased workload imposed on the human pilot remains within what they can

accomplish. Especially because of the different ways in which this architecture option increases the workload required of the human pilot, any assumptions made about the level of workload imposed must be carefully verified to ensure that they accurately reflect what human pilots will actually experience in the real system.

In contrast to this architecture, an alternative architecture option could be considered that lowers the workload of the human pilot by assigning more responsibilities to the ASEC. An example of such an architecture will be explored next in architecture option 2.

4.6.4 Architecture Option 2: ASEC Proposes Flight Trajectories for Human Pilot to Execute

As identified in the analysis of option 1, one major challenge is the high workload imposed on the human pilot by having most of the responsibilities for flight in DVEs assigned to them and the analysis identified three sources that of high workload:

1. Increased cognitive effort required to validate feedback and update their mental model
2. Difficult decisions that either need to be made quickly and accurately or that are made under uncertainty or incomplete information
3. The increased size of the mental model that the human pilot must maintain and keep updated

Therefore, one possible way to alleviate the high workload imposed on the human pilot could be to address the first two sources of that high workload. That is, to increase the ASEC's involvement in some responsibilities to help the human pilot validate feedback and update their mental model as well as consider alternatives when making decisions under difficult conditions. For these reasons, this architecture option makes use of a moderate level of automation by increasing the ASEC's involvement in the responsibilities necessary to safely fly the aircraft. As will be shown, although the workload imposed on the human pilot might be lower, there are tradeoffs associated with the use of more automation in the system.

For compactness, only the process model parts and responsibilities that have different assignments than those shown in architecture option 1 will be shown. Any process model part or responsibility that has the same assignment in both architecture options is not repeated.

Assignment of Process Model Parts

In this architecture option, only two of the process model parts have different assignments compared to option 1. With the increased involvement of the ASEC, the ASEC must now maintain a copy of the anticipated future state of the airspace part of the process model, just like the human pilot would. Consequently, the ASEC must also maintain a copy of the model of system behavior as well since it needs that information to carry out some of the responsibilities that it now shares with the human pilot. As a result, the anticipated future state of the airspace along with the model of system behavior parts of the process model are now shared between the human pilot and ASEC as shown in Table 34 instead of just being assigned to the human pilot.

Table 34: Assignment of Process Model Parts for Part 2 Architecture Option 2

Process Model	Assigned to		Rationale/Assumptions
	Pilot	ASEC	
Anticipated Future Airspace State	X	X	Most of this information is generated automatically and can be easily parsed by automation. However, the human pilot must share this process model part not only because they need it to fly the aircraft but also so that they can account for more complex feedback that they ASEC cannot account for. This assumes some basic kinematic models are available to project future airspace state but requires pilot assistance to account for more complex intent information.
Model of System Behavior	X	X	Since the ASEC is responsible for much of the detection responsibilities, the ASEC and pilot both need this part of the process model to interpret sensor and other data appropriately

Assignment of Non-Feedback-Validation Responsibilities

As described at the beginning of this architecture option, the workload imposed on the human pilot could be reduced by providing the human pilot with assistance for responsibilities that involve decision-making under difficult conditions. Based on the system-level behavior information, it can be observed that SR-16, SR-17, SR-20, SR-23 and SR-24 are responsibilities of this type and therefore sharing them between the human pilot and the ASEC instead of having the human pilot alone assigned to them might be able to help the human pilot overcome human cognitive limitations and consider a broader array of factors or alternatives before making. decision. These assignments are shown in Table 35 along with the underlying rationale and assumptions associated with each assignment.

Table 35: Assignment of Non-Feedback-Validation Responsibilities for Part 2 Architecture Option 2

Resp. ID	Responsibility	Assigned to		Rationale/Assumptions
		Pilot	ASEC	
SR-16	Account for the current and future movements of other aircraft in the vicinity when selecting actuator movements [SLR-17]	X	X	In this example, the ASEC is designed to help the human pilot to find acceptable flight paths/trajectories and accounting for objects and other aircraft in the surrounding airspace. The human pilot shares this responsibility because the human pilot checks and may select an alternative path if they decide the ASEC-selected one is not appropriate.
SR-17	Ensure that aircraft movements are selected such that sufficient reaction time is available if intent or movements of other aircraft are not what was expected, even if no violation of minimum separation is initially expected [SLR-18]	X	X	Similar to SR-16, the ASEC is designed to help the human pilot to find acceptable flight paths/trajectories including ensuring that the trajectory it selects incorporates sufficient reaction time to change flight path if nearby aircraft move in unexpected ways. The human pilot shares this responsibility because the human pilot checks and may select an alternative path if they decide the ASEC-selected one is not appropriate.
SR-20	Select actuator movements that minimize the risk of violation of minimum separation when information about the state of the airspace is in a degraded condition (e.g. delayed) [SLR-21]	X	X	Similar to SR-16, the ASEC is designed to help the human pilot to find acceptable flight paths/trajectories including ensuring that the flight path it selects minimizes the risk of violation of minimum separation. The human pilot shares this responsibility because the human pilot checks and may select an alternative path if they decide the ASEC-selected one is not appropriate.
SR-23	Ensure that a viable flight path is always available to avoid any violations of minimum separation [SLR-24]	X	X	Similar to SR-16, the ASEC is designed to help the human pilot to find acceptable flight paths/trajectories including avoiding all possible violations of minimum separation. The human pilot shares this responsibility because the human pilot checks and may select an alternative path if they decide the ASEC-selected one is not appropriate.
SR-24	Select a viable flight path that avoids all possible violations of minimum separation [SLR-26]	X	X	

Assignment of Feedback Validation Responsibilities

In addition to sharing some of the non-feedback-validation responsibilities between the human pilot and ASEC, the ASEC can also share more of the feedback validation responsibilities as well. This has the benefit of potentially reducing the cognitive effort needed to validate feedback and update mental models. For some responsibilities and process model parts, assignments are transferred from the human pilot to the ASEC while for others, they become shared with the ASEC. Some examples of these responsibility assignments are shown in Table 36.

Table 36: Assignment of Feedback Validation Responsibilities for Part 2 Architecture Option 2

<table>
<tr><th>Resp ID</th><th>Responsibility</th><th>Assignments for Each Process Model Part</th><th>Rationale/Assumptions</th></tr>
<tr><td>SR-2</td><td>Account for prevailing DVE conditions and their possible effect on feedback sources when making use of feedback information [SLR-2]</td><td><table>
<tr><td>Aircraft Mission Readiness</td><td>N/A</td></tr>
<tr><td>Current Aircraft System State</td><td>N/A</td></tr>
<tr><td>Current Aircraft Navigation State</td><td>ASEC</td></tr>
<tr><td>Expected Safe Departure State</td><td>N/A</td></tr>
<tr><td>Current Weather Conditions</td><td>ASEC</td></tr>
<tr><td>Current Airspace State</td><td>ASEC</td></tr>
<tr><td>Anticipated Future Airspace State</td><td>Pilot ASEC</td></tr>
<tr><td>Model of System Behavior</td><td>N/A</td></tr>
</table></td><td>For the anticipated future airspace state, the human pilot shares this responsibility because both the ASEC and human pilot may receive feedback for this part of the process model and will have to account for the effects of DVE when they use that feedback they may receive

Assumes that the determinations involved are easily made based on simple interpretations of the DVE conditions and that complex logic is not required</td></tr>
<tr><td>SR-3</td><td>Validate the inputs or feedback received from other controllers before using that input or feedback to update process models [SLR-3]</td><td><table>
<tr><td>Aircraft Mission Readiness</td><td>Pilot</td></tr>
<tr><td>Current Aircraft System State</td><td>ASEC</td></tr>
<tr><td>Current Aircraft Navigation State</td><td>ASEC</td></tr>
<tr><td>Expected Safe Departure State</td><td>N/A</td></tr>
<tr><td>Current Weather Conditions</td><td>ASEC</td></tr>
<tr><td>Current Airspace State</td><td>ASEC</td></tr>
<tr><td>Anticipated Future Airspace State</td><td>Pilot ASEC</td></tr>
<tr><td>Model of System Behavior</td><td>N/A</td></tr>
</table></td><td>Except for mission readiness, expected safe departure state, anticipated future airspace state and model of system behavior, this option assumes that the ASEC can automatically validate inputs for the other process model parts.

For the anticipated future airspace state, the human pilot shares this responsibility because they may receive information in a format that is not easily interpreted by automation (e.g. voice over radio) and will therefore also be involved in validating those inputs</td></tr>
</table>

<table>
<tr><th>Resp ID</th><th>Responsibility</th><th>Assignments for Each Process Model Part</th><th>Rationale/Assumptions</th></tr>
<tr><td>SR-8</td><td>Process all feedback to make a deliberate decision if it is to be ignored/dropped [SLR-8]</td><td><table>
<tr><td>Aircraft Mission Readiness</td><td>Pilot</td></tr>
<tr><td>Current Aircraft System State</td><td>ASEC</td></tr>
<tr><td>Current Aircraft Navigation State</td><td>ASEC</td></tr>
<tr><td>Expected Safe Departure State</td><td>Pilot</td></tr>
<tr><td>Current Weather Conditions</td><td>ASEC</td></tr>
<tr><td>Current Airspace State</td><td>ASEC</td></tr>
<tr><td>Anticipated Future Airspace State</td><td>Pilot
ASEC</td></tr>
<tr><td>Model of System Behavior</td><td>Pilot</td></tr>
</table></td><td>Except for mission readiness, expected safe departure state, anticipated future airspace state and model of system behavior, this option assumes that the ASEC can automatically validate inputs for the other process model parts.

For the anticipated future airspace state, the human pilot shares this responsibility because they may receive information in a format that is not easily interpreted by automation (e.g. voice over radio) and will therefore also be involved in validating those inputs</td></tr>
<tr><td>SR-14</td><td>Process sensor data and use it to update its process model sufficiently quickly [SLR-14]</td><td><table>
<tr><td>Aircraft Mission Readiness</td><td>Pilot</td></tr>
<tr><td>Current Aircraft System State</td><td>ASEC</td></tr>
<tr><td>Current Aircraft Navigation State</td><td>ASEC</td></tr>
<tr><td>Expected Safe Departure State</td><td>N/A</td></tr>
<tr><td>Current Weather Conditions</td><td>ASEC</td></tr>
<tr><td>Current Airspace State</td><td>ASEC</td></tr>
<tr><td>Anticipated Future Airspace State</td><td>Pilot
ASEC</td></tr>
<tr><td>Model of System Behavior</td><td>N/A</td></tr>
</table></td><td>For everything but the aircraft mission readiness, anticipated future airspace state and model of system behavior, the ASEC is primarily the one updating the process model parts.

However, in the case of anticipated future airspace state, the human pilot shares the responsibility because some feedback is more easily interpreted by the human pilot rather than the ASEC.

For aircraft mission readiness, the feedback for those process model parts are still most easily interpreted by a human than by automation.

Expected safe departure state and model of system behavior are marked as N/A because they were not assigned to the Piloting Controller in part 1</td></tr>
</table>

Architecture Option 2 Control Structure

The resulting control structure for this architecture option is shown in Figure 23.

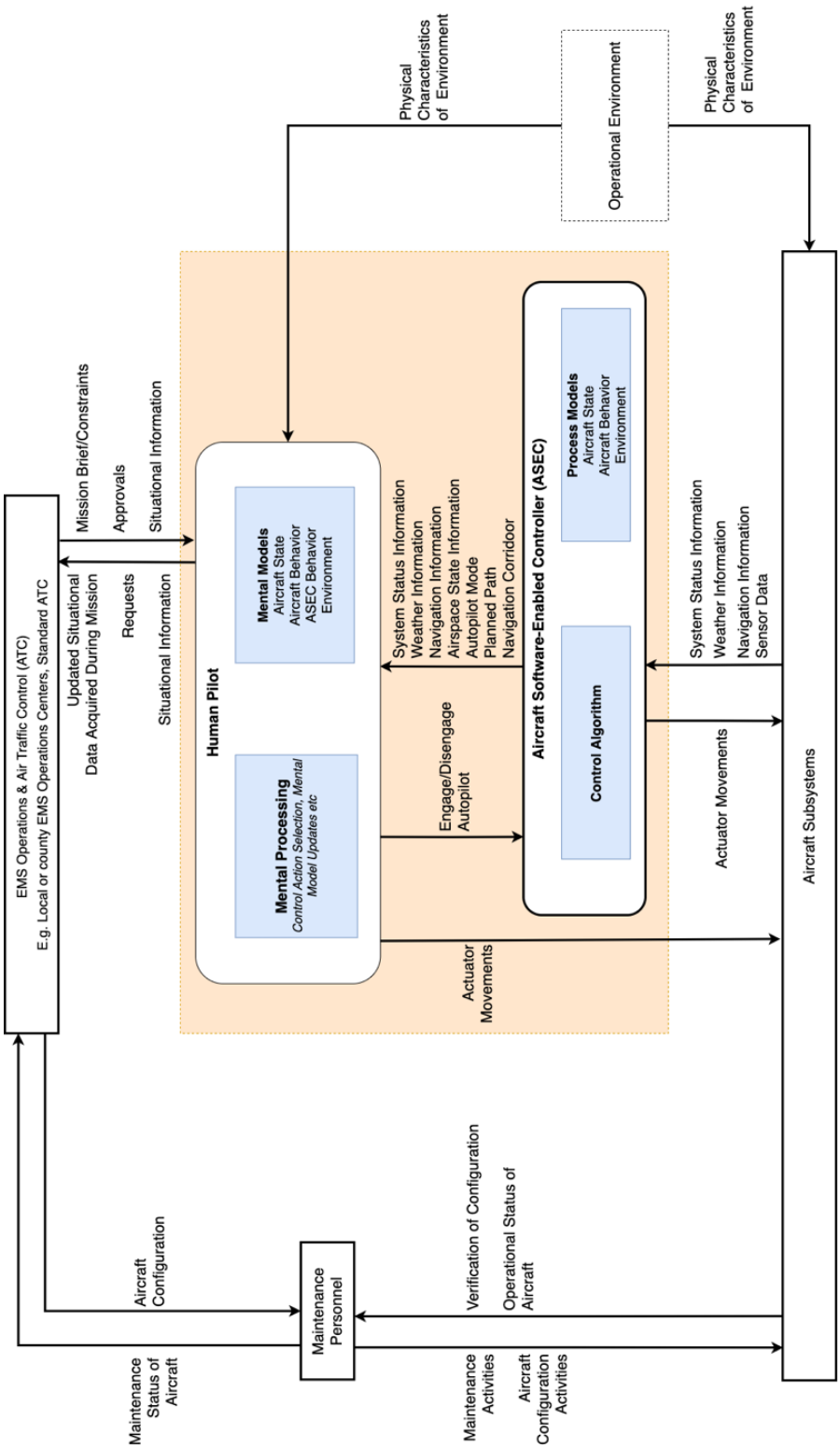


Figure 23: Part 2 Architecture Option 2 Control Structure

4.6.5 Architecture Analysis of Option 2

One of the goals of this architecture was to alleviate the potentially high workload imposed on the human pilot if the human pilot is primarily responsible for controlling the aircraft and the ASEC simply serves as a decision aid. Although this goal appears to be achieved because the ASEC is assigned more responsibilities in this architecture option compared to option 1, this benefit has accompanying tradeoffs that must be analyzed to determine their impact on the architecture and how difficult it will be to implement this architecture during detailed system design. This analysis is important to ensure that the potential benefits can be compared to the costs incurred when deciding whether to select this architecture option to move forward with.

The main benefit of this architecture option is that it reduces the workload imposed on the human pilot by assigning more responsibilities to the ASEC compared to option 1. For example, as shown in the previous section, many of the feedback validation responsibilities are assigned to the ASEC instead of the human pilot, potentially alleviating some of the high workload observed with option 1. Table 37 shows the feedback validation responsibility assignments from Table 36 but presents them in a slightly different format to illustrate the increased involvement of the ASEC compared to the human pilot in carrying out these responsibilities.

Table 37 shows that for this architecture option, the ASEC is assigned in many more of the cells in the table as compared to option 1 where the human pilot was assigned in most of them. For example, all four of the feedback validation responsibilities are assigned to the ASEC to carry out for the current aircraft system state, current aircraft navigation state, current weather conditions and current airspace state. By comparison, the human pilot is only assigned feedback validation responsibilities for the aircraft mission readiness, expected safe departure state, anticipated future airspace state and model of system behavior parts of the process model. By sharing more of these responsibilities with the ASEC, the workload imposed on the human pilot has the potential to be reduced because the human pilot is assigned a smaller fraction of these responsibilities compared to option 1. In addition, for the responsibilities that the human pilot is assigned to, the rate at which those responsibilities need to be carried out are relatively low as defined in the timing requirements part of the system-level behavior. So, with fewer assigned responsibilities that only need to be carried out at relatively low rates, the workload on the human pilot should be reduced.

Table 37: Sample of Feedback Validation Responsibilities Showing Increased Involvement of ASEC

		Responsibilities			
		SR-2	SR-3	SR-8	SR-14
		Account for prevailing DVE conditions and their possible effect on feedback sources when making use of feedback information [SLR-2]	Validate the inputs or feedback received from other controllers before using that input or feedback to update process models [SLR-3]	Process all feedback to make a deliberate decision if it is to be ignored/dropped [SLR-8]	Process sensor data and use it to update its process model sufficiently quickly [SLR-14]
Process Model Parts	Aircraft Mission Readiness	N/A	Human Pilot	Human Pilot	Human Pilot
	Current Aircraft System State	N/A	ASEC	ASEC	ASEC
	Current Aircraft Navigation State	ASEC	ASEC	ASEC	ASEC
	Expected Safe Departure State	N/A	N/A	Human Pilot	N/A
	Current Weather Conditions	ASEC	ASEC	ASEC	ASEC
	Current Airspace State	ASEC	ASEC	ASEC	ASEC
	Anticipated Future Airspace State	Shared between Human Pilot and ASEC	Shared between Human Pilot and ASEC	Shared between Human Pilot and ASEC	Shared between Human Pilot and ASEC
	Model of System Behavior	Human Pilot	N/A	Human Pilot	N/A

One challenge with assigning more responsibilities to the ASEC instead of the human pilot is that it is even more important that sufficient constraints are placed on the behavior of the system and the ASEC software to avoid unsafe behavior. This is because many of the challenges related to the design of the feedback mechanisms in option 1 are still applicable in this architecture option as poorly designed feedback mechanisms can still lead to unsafe behavior even if the ASEC software is receiving and interpreting that feedback instead of a human pilot. Furthermore, the ASEC software can behave in similar unsafe ways as a human pilot if human decision-making biases are encoded into the software. This can be illustrated using responsibility SR-2 and its associated causal scenario.

Responsibility SR-2: Account for prevailing DVE conditions and their possible effect on feedback sources when making use of feedback information [SLR-2]

Loss Scenario CS-1.1.1-1.1.2: The piloting controller believes it is receiving conflicting feedback because DVE conditions have degraded the accuracy of or obscured some (but not all) feedback sources. The piloting controller therefore chooses to ignore the conflicting feedback, believing that it is a false positive or erroneous. The piloting controller is especially susceptible to this if the majority of sources appear to agree that no aircraft is present and fewer sources appear to show otherwise. As a result, the piloting controller updates its process model based only on what it wrongly believes to be the correct feedback. [SLR-2]

Even when SR-2 is assigned to the ASEC instead of to the human pilot, the ASEC can still wrongly believe it is receiving conflicting feedback for the same reasons that a human pilot might do so. For example, if the ASEC is initially presented with feedback that does not show an aircraft nearby, even if an aircraft is subsequently detected, it may be more inclined to ignore that detection if its algorithm relies on assumptions about when an aircraft should be detected. Alternatively, if the ASEC software uses a voting system to decide how to resolve conflicting feedback, such a voting system may wrongly decide that no aircraft or object exists nearby if most of the sensors do not detect the aircraft or object and only a few are able to correctly detect the nearby aircraft or object. These unsafe behaviors of the ASEC software can occur if the software engineers base their software design on the same decision-making biases that a human pilot might use. As a result, similar types of decision-making biases can lead to unsafe system behavior regardless of whether the human pilot or ASEC is assigned to carry out the responsibility. For these reasons, although assigning more of these data integrity responsibilities to the ASEC reduces the workload on the human pilot, it is even more important that sufficient constraints are placed on the behavior of the system and software which can lead to additional difficulties in designing and implementing the system.

Another challenging aspect of this architecture option is that the sharing of responsibilities can increase the complexity of the system, making the system more difficult to design and implement successfully. This is because when responsibilities are shared between the human pilot and ASEC, adequate coordination is required to ensure that they carry out their shared responsibilities effectively. As a result, additional requirements or constraints must be imposed on system behavior to ensure that adequate coordination is achieved. To understand how the complexity of the system may increase due to the need for adequate coordination, the STPA extension for coordination developed by Kip Johnson [15] can be used to analyze this architecture option and there are two negative outcomes for this architecture that will need to be prevented.

The first negative outcome if there is inadequate coordination between the human pilot and ASEC is that some of the associated loss scenarios become more difficult to mitigate or prevent. This is because inadequate coordination between the human pilot and ASEC can result in new ways that existing loss scenarios might occur. The loss scenarios associated with these responsibilities must therefore be updated to identify the additional ways that inadequate coordination between the ASEC and the human pilot can lead to unsafe behavior so that

additional constraints can be placed on the system's behavior. This can be illustrated using SR-2 that is shared between the human pilot and ASEC for maintaining the Anticipated Future State of the Airspace process model part as shown in Table 37.

When SR-2 is shared between the pilot and ASEC, the human pilot and ASEC may disagree on how feedback should be used to update their shared process model of the future state of the airspace and there are several reasons this could occur. Some examples include:

- The human pilot and ASEC interpret feedback differently or use it in different ways to update the process model of future airspace state.
- The human pilot might have access to additional feedback (e.g. verbal radio-based communication) that the ASEC does not

As a result of any of these reasons, the ASEC and human pilot now have different beliefs of what the anticipated future state of the airspace will be and they will need to coordinate to resolve their conflicting proposals. However, if the human pilot or ASEC have inadequate ways to communicate how they each used the available feedback to update their process model, then, as Johnson identified in [15], such a condition could lead to inadequate coordination. Alternatively, if the human pilot has no way to share additional feedback (e.g. verbal radio communications) that they received with the ASEC, then the human pilot and ASEC are not using the same set of feedback to update the process model of the future state of the airspace. As Johnson identified in [15], inadequate observation of common objects is another condition that could lead to inadequate coordination between the human pilot and ASEC. If there is inadequate coordination between the human pilot and the ASEC, the human pilot could independently choose to wrongly ignore the ASEC's assessment of the future state of the airspace, even if the ASEC's assessment matches the actual future state of the airspace and the human pilot's assessment does not. This scenario is made even more likely if the human pilot has a low level of trust in the ASEC. This example thus illustrates how the need for adequate coordination can result in scenarios that are harder to mitigate because of possible coordination problems, thus requiring additional requirements to be added to prevent inadequate coordination from occurring.

The second negative outcome if there is inadequate coordination between the human pilot and ASEC is that although the shared responsibilities should reduce the workload imposed on the human pilot, the anticipated workload savings may not be realized if the human pilot and ASEC are unable to coordinate effectively to carry out their shared responsibilities. This may happen because when responsibilities are shared, the human pilot and ASEC must work together for their shared responsibilities to be carried out effectively. However, if coordination between the human pilot and ASEC is missing or insufficient, the human pilot and ASEC will not be able to work together effectively. As a result, not only will the safety constraints not be adequately enforced but the need to coordinate with the ASEC may also impose significant amounts of additional workload on the human pilot. Consequently, the benefits of sharing responsibilities may be negated or the overall workload experienced by the human pilot may even increase. To illustrate how inadequate coordination may negate the benefits of shared responsibilities, SR-16 and its associated loss scenarios (Table 38) can be analyzed as an example.

Table 38: SR-16 and its Associated Loss Scenarios

Resp. ID	Responsibility	Scenario ID	Scenario
SR-16	Account for the current and future movements of other aircraft in the vicinity when selecting actuator movements [SLR-17]	CS-1.2.1-2	The piloting controller provides actuator movements despite receiving feedback that there was another aircraft or object nearby because the piloting controller may have the wrong belief about the future behavior of the other aircraft or object and therefore believes that providing actuator movements will not pilot the aircraft toward the other aircraft or object [SLR-17, SLR-18]
		CS-1.2.1-3	The piloting controller provides actuator movements despite receiving feedback that there was another aircraft or object nearby because the piloting controller is forced to make a quick decision to avoid violating minimum separation that does not fully account for all objects and aircraft in the airspace. As a result, the piloting controller tries to avoid one object/aircraft and collides with another instead [SLR-17, SLR-18]

The loss scenarios show that for SR-16 to be carried out effectively, the piloting controller as a whole must have an accurate understanding of the future behavior of other aircraft and objects and must be able to consider all possible violations of minimum separation when selecting actuator movements. Sharing SR-16 between the human pilot and ASEC may therefore be beneficial because the ASEC can assist the human pilot in determining the future behavior of other aircraft or objects or assist the human pilot in accounting for all possible violations of minimum separation when selecting actuator movements.

However, by sharing this responsibility, these benefits can only be attained if the human pilot and ASEC can coordinate and work together well. If the human pilot does not coordinate at all with the ASEC when selecting actuator movements, then the human pilot becomes susceptible to the same loss scenarios involving high workload and human decision-making biases that were discussed in option 1. As described in [15], reasons that the human pilot may choose not to coordinate with the ASEC at all include a lack of trust in the ASEC or having insufficient time or resources available to perform adequate coordination. As a result of any of these reasons, the human pilot may choose to make an independent decision and ignore the ASEC.

Unsafe behavior may also occur when coordination between the human pilot and ASEC is present but inadequate and multiple factors can contribute to inadequate coordination occurring. For

example, for SR-16, if the method by which future movements of other aircraft should be accounted for is ambiguous or the criteria for a trajectory to be deemed acceptable is not clearly defined, the human pilot and ASEC may employ different methods to account for the current and future movements of other aircraft and objects in the vicinity. They may also make use of different assumptions in their decision making or their use of feedback when selecting actuator movements. As a result, the human pilot and ASEC might propose different sets of actuator movements and the human pilot will have to resolve these differences before executing the chosen set of actuator movements. To do this, the human pilot will need to be able to both understand how the ASEC selected its proposed actuator movements and communicate to the ASEC how they selected their proposed actuator movements. As described in [15], if there is inadequate communication available, it may be difficult or impossible for the human pilot and ASEC to understand each other and reach a common consensus on what the actuator movements should be. Alternatively, if the framework for resolving the conflicting proposals is not clearly defined, the human pilot may have trouble deciding how to combine the two proposed sets of actuator movements. As a result of factors such as these, the human pilot may have difficulties coordinating with the ASEC. Not only does this result in a higher workload than anticipated but the human pilot may also be delayed in providing the necessary actuator movements while they try to coordinate with the ASEC. As such, this example illustrates how inadequate coordination can not only lead to a negation of the desired benefits but also lead to new causal scenarios that will need to be mitigated by introducing additional system requirements and constraints.

In summary, this architecture option primarily aims to reduce the workload of the human pilot by transferring or sharing more responsibilities with the ASEC. However, this benefit comes with two costs. The first is that it is even more important to place sufficient constraints on the behavior of the system and software to avoid unsafe behavior and the second is that the complexity of the system may increase as additional requirements are added to ensure adequate coordination between the human pilot and ASEC and prevent scenarios involving inadequate coordination from occurring. It is therefore important to compare these costs to the benefits obtained to help determine if the benefits are worth the costs incurred if this architecture were selected.

4.6.6 Architecture Option 3: Human Pilot Supervises Automated ASEC's Control of Flight

One final architecture option is an architecture where the ASEC is also given control of the low-level flight controls and the human pilot begins to resemble more of a supervisor over the flight of the aircraft rather than being directly involved in the aircraft flight itself. One reason for doing this is that when humans are required to control a machine such as an aircraft, their ability to perform fast and accurate control is affected by a principle known as the speed-accuracy tradeoff [30]. This principle essentially states that there is a negative correlation between the speed and accuracy with which humans can control a system such that if a human operator must carry out a series of actions quickly, they are more likely to make errors. By this principle, if flight under DVE conditions requires the human pilot to make changes to their flight controls quickly to respond to changing DVE conditions, such a control task may be challenging for a human pilot to perform without making an error of judgement when selecting actuator movements. For this

reason, an architecture option where the ASEC is given control of the low-level flight controls instead of the human pilot could be worth evaluating.

As before, only the differences in the process model part or responsibility assignments between architecture options 2 and 3 are shown and any that have the same assignments in options 2 and 3 are not repeated.

Assignment of Process Model Parts

The first difference is in one of the assignments of process model parts. As shown in Table 39, in this architecture option, the current weather conditions process model part is only assigned to the ASEC unlike in option 2, where it is shared between the human pilot and ASEC.

Table 39: Assignment of Process Model Parts for Part 2 Architecture Option 3

Process Model	Assigned to		Rationale/Assumptions
	Pilot	ASEC	
Current Weather Conditions		X	Much of this information is generated automatically and can be easily parsed by automation. Since the ASEC is now handling low-level flight controls, this part of the process model no longer needs to be shared with the human pilot

Assignment of Non-Feedback-Validation Responsibilities

The other two changes are to the assignments of non-feedback-validation responsibilities. Unlike in option 2 where the human pilot directly controlled the aircraft, the human pilot is no longer in direct control of the aircraft in this architecture option and therefore SR-21 and SR-22 are now only assigned to the ASEC as shown in Table 40.

Table 40: Assignment of Non-Feedback-Validation Responsibilities for Part 2 Architecture Option 3

Resp. ID	Responsibility	Assigned to		Rationale/Assumptions
		Pilot	ASEC	
SR-21	Respond quickly and with appropriate magnitude to disturbances to prevent unintended movement of the aircraft [SLR-22]		X	Assumes that the automation is now capable of stabilizing the aircraft and maintaining its current speed, heading and position under all DVE conditions without intervention from the human pilot
SR-22	Respond quickly enough and with appropriate magnitude to select and effect the desired change in flight path under the given environmental conditions [SLR-23, SLR-25]		X	Assumes that the automation is now capable of effecting trajectories under all DVE conditions without intervention from the human pilot

Assignment of Feedback Validation Responsibilities

This architecture option makes no changes to the assignments of feedback validation responsibilities compared to option 2. Therefore, no additional information will be shown here for these assignments.

Architecture Option 3 Control Structure

The resulting control structure for this architecture option is shown in Figure 24.

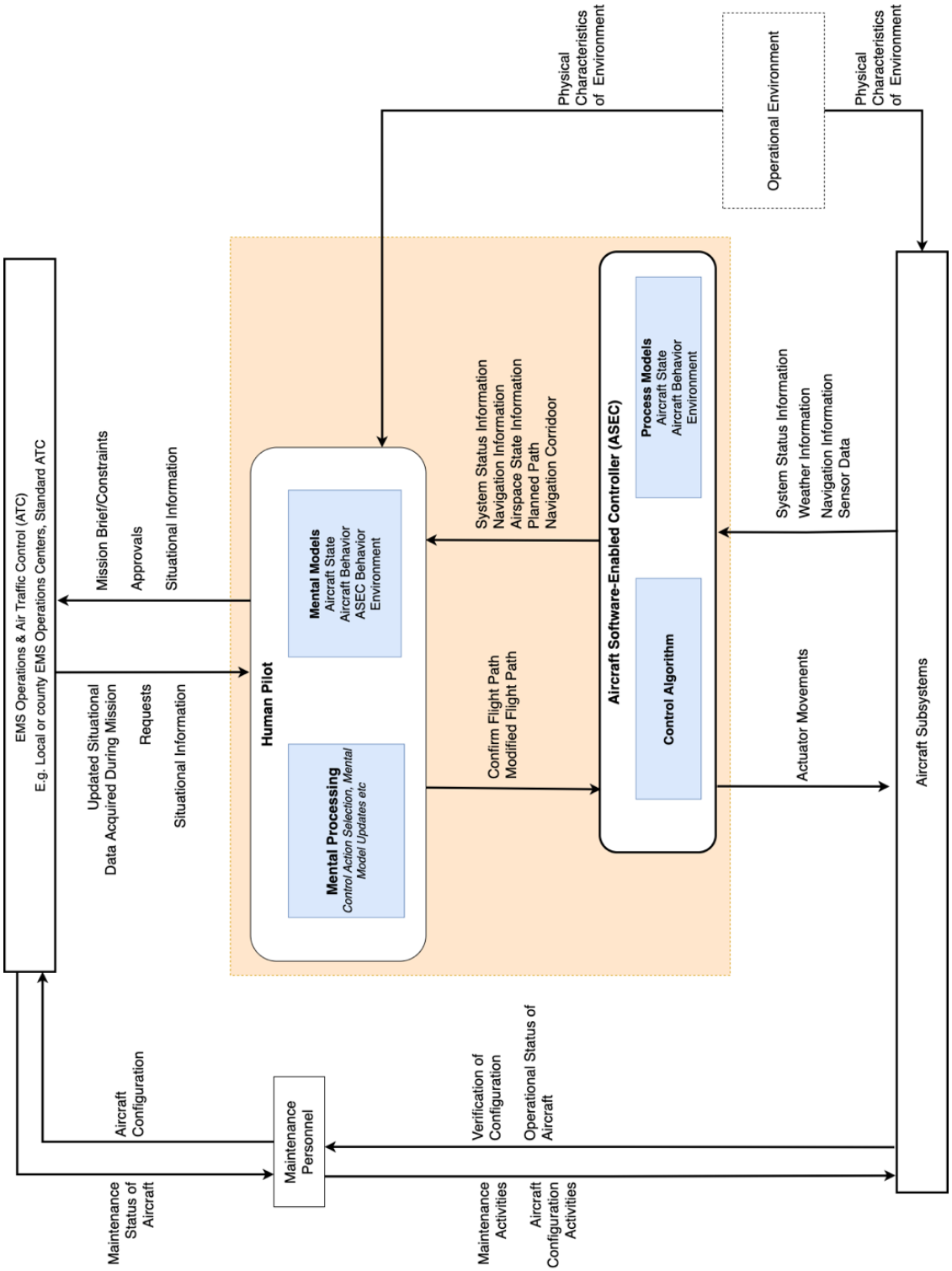


Figure 24: Part 2 Architecture Option 3 Control Structure

4.6.7 Architecture Analysis of Option 3

As discussed in the introduction to this architecture option, the main goal of this architecture option was to further reduce the workload of the human pilot by alleviating them of the need to manipulate low-level flight controls. This would allow the human pilot to focus on higher-level, more complex and ambiguous decision making that humans are better at performing. As with architecture option 2, although this goal appears to be achieved since the human pilot no longer needs to maintain a mental model of the weather conditions and is no longer assigned to two responsibilities for flying the aircraft, these changes result in a significantly different relationship between the human pilot and ASEC. Therefore, the modified architecture should be fully analyzed to determine how the loss scenarios and the constraints necessary to prevent unsafe behavior change in this architecture.

The main reason that this architecture option results in a significant change to the architecture of the Piloting Controller is because it represents a fundamentally different relationship between the human pilot and the ASEC. In architecture options 1 and 2, the human pilot always retained the responsibilities for flying the aircraft and the ASEC was being assigned more of the other responsibilities necessary for safe flight in DVEs. In essence, the role of the human pilot in the architecture remained that of a system operator in architecture options 1 and 2. However, in this architecture option, by assigning the responsibilities for flying the aircraft to the ASEC instead of the human pilot, the role of the human pilot now begins to resemble more of a supervisory role instead of a system operator. Consequently, by comparing Figure 23 and Figure 24, it can be observed that the control structure changes in two important ways. Firstly, the human pilot does not provide actuator movements to the aircraft subsystems anymore and no longer needs to engage an “Autopilot” function because the ASEC is now always flying the aircraft. However, since the human pilot and ASEC must still be able to work together and coordinate to carry out the other responsibilities that they share, the human pilot must be able to influence the ASEC’s choice of flight trajectory and actuator movements that it provides. As a result, the second change to the control structure is that, although the human pilot is no longer flying the aircraft directly, they can influence the ASEC’s behavior by confirming or modifying the ASEC’s proposed flight paths. This is illustrated in Figure 25 where the new control actions between the human pilot and ASEC are circled in red.

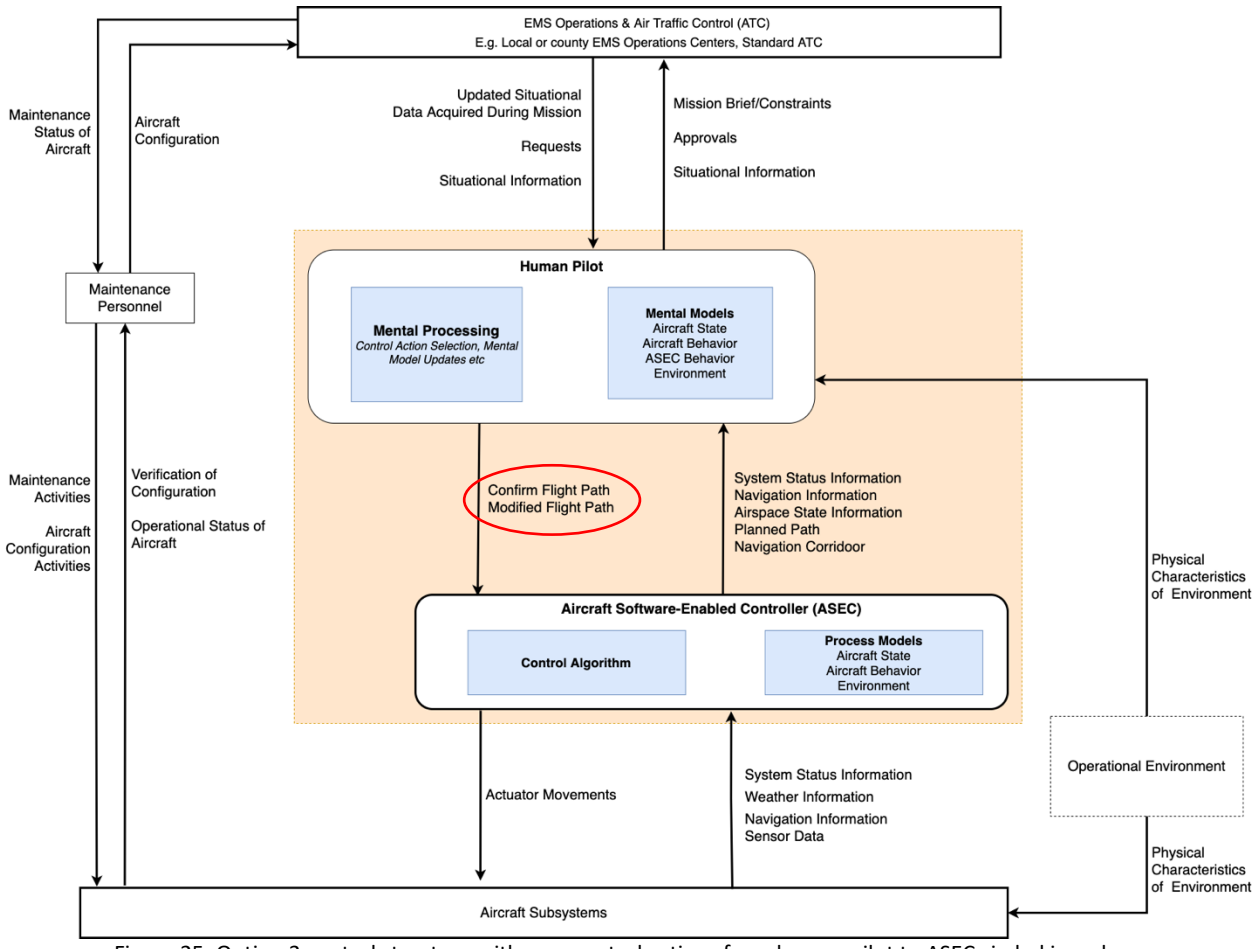


Figure 25: Option 3 control structure with new control actions from human pilot to ASEC circled in red

As a result of these changes to the control structure, a revised STPA analysis of this system needs to be performed to identify the new loss scenarios that arise. Two aspects of this revised STPA analysis will be discussed in this architecture analysis. Firstly, new scenarios will need to be identified for the ASEC's "Actuator Movements" control action because the decision by the ASEC to provide or not provide Actuator Movements to the aircraft subsystems can now also be influenced by inputs from the human pilot in addition to decision-making algorithms contained within the ASEC software. The second aspect that will be discussed are the new UCAs and scenarios for the new control actions from the human pilot to the ASEC that are highlighted in Figure 25. Since these control actions are new to the control structure, they must be analyzed using STPA to identify the UCAs and loss scenarios that will need to be mitigated so that additional safety constraints can be imposed on the system to prevent unsafe system behavior from occurring. These revisions to the STPA analysis for this system are important to perform because the analysis results and the additional system requirements that are identified can be used to inform whether the benefits of this architecture option are worth the tradeoffs in terms of the added complexity needed to prevent unsafe behavior in this system.

Revised STPA Analysis of ASEC Actuator Movements Control Action

Beginning with the Actuator Movements control action between the ASEC and Aircraft Subsystems, the UCAs for this control action are essentially the same as those identified in the original STPA analysis for the Piloting Controller and therefore those UCAs do not change. However, additional loss scenarios in addition to those already identified should be generated that consider in more detail how the ASEC could issue unsafe control actions. Using UCA-1.2 modified for the ASEC instead of the piloting controller, some example new scenarios that are identified for the first basic scenario type (unsafe controller behavior) are shown in Table 41 and more examples can be found in Appendix G.

UCA-1.2: ASEC provides actuator movements that steers the aircraft toward another aircraft or object

Table 41: Example Additional Loss Scenarios Resulting from Unsafe Controller Behavior for UCA-1.2

Scenario ID	Scenario
CS-G-1.2.1-1.1	Despite receiving feedback that there was another aircraft or object nearby, the ASEC may have the wrong process model of the state of the environment, the aircraft or the airspace around the aircraft and therefore wrongly believes that the nearby airspace is clear to pilot toward. Although the ASEC coordinates with the human pilot to confirm the trajectory that it planned, the erroneous trajectory planned by the ASEC is confirmed without modification. The ASEC might receive confirmation of its erroneously planned trajectory because the human pilot and ASEC might have the same shared process model of the current state of the airspace. As a result, the human pilot also does not recognize that the planned trajectory is headed toward a nearby object or aircraft and therefore does not correct the trajectory.
CS-G-1.2.1-2	Despite receiving feedback that there was another aircraft or object nearby, both the human and the ASEC have wrong process models of the state of the environment, the aircraft or the airspace around the aircraft. However, their process models are wrong in different ways. As a result, the human pilot might still correctly recognize the error in the ASEC's planned trajectory. However, although the modifications that the human pilot makes addresses the error that the ASEC made, the human pilot's modified trajectory now pilots the aircraft toward a different aircraft or object in the airspace instead.
CS-G-1.2.1-3	Despite receiving feedback that there was another aircraft or object nearby, the ASEC has the wrong process model of the state of the environment, the aircraft or the airspace around the aircraft and therefore wrongly believes that the nearby airspace is clear to pilot toward. Although the human pilot does recognize the error in the ASEC's planned trajectory, the human pilot may be delayed in providing a confirmation or a modification to the flight path to the ASEC. If the ASEC has a timeout programmed into it, it may wrongly decide that when the timeout has expired, it should simply execute its incorrect proposal without waiting for the human pilot.

STPA Analysis of Human Pilot Control Actions

Similarly, the new control actions issued by the human pilot can be analyzed using STPA. As shown in Figure 25, the human pilot has two control actions that it can issue to the ASEC: Confirm Flight Path and Modify Flight Path.

Each of these control actions should therefore be analyzed to determine how the human pilot might issue an unsafe control action that could lead to a hazard or a loss. Table 42 shows some example UCAs for the "Confirm Flight Path" and "Modify Flight Path" control actions and more examples can be found in Appendix G.

Table 42: Example UCAs for Human Pilot New Control Actions

Control Action	Providing	Not Providing	Provide too early/too late	Applied too long/Stopped too soon
Confirm Flight Path	UCA-3.1: Pilot confirms flight path when that flight path pilots the aircraft into another aircraft or object UCA-3.3: Pilot confirms flight path that exceeds the bounds of the mission parameters (e.g. exceeds max altitude or operational area)	UCA-3.4: Pilot does not confirm flight path when a new flight path is needed to avoid piloting the aircraft into another aircraft or object	UCA-3.6: Pilot confirms the flight path too late after the last opportunity to avoid violation of minimum separation has passed	N/A
Modify Flight Path	UCA-3.8: Pilot provides modified flight path when the modified flight path will pilot the aircraft toward the other aircraft or object	UCA-3.11: Pilot does not provide modified flight path when the path needs to be modified to prevent violation of minimum separation	UCA-3.13: Pilot provides modified flight path too late after the aircraft begins to execute the unmodified new flight path that will result in a violation of minimum separation	N/A

Each of these UCAs can then be analyzed to identify the loss scenarios that could lead to these UCAs. Two examples are shown in Table 43 for UCA-3.1 and UCA-3.8 and more example scenarios can be found in Appendix G.

UCA-3.1: Pilot confirms flight path when that flight path pilots the aircraft into another aircraft or object

UCA-3.8: Pilot provides modified flight path when the modified flight path will pilot the aircraft toward the other aircraft or object

Table 43: Example Scenarios for UCA-1 and UCA-8

UCA ID	Scenario ID	Scenario
UCA-3.1	CS-G-3.1.1-1	Pilot confirms the flight path despite receiving feedback that the flight path pilots the aircraft into another aircraft or object because, under a period of high workload or stress, the human pilot does not update their mental model of the state of the airspace (e.g. due to limited attention resources or cognitive tunneling) and uses their inaccurate mental model to evaluate the flight path instead. As a result, the human pilot does not realize that the flight path proposed by the ASEC pilots the aircraft into another aircraft or object and confirms it, wrongly believing that the flight path does not pilot the aircraft toward another aircraft or object.
UCA-3.8	CS-G-3.8.1-1	The human pilot provides a modified flight path despite receiving feedback that the modified flight path will pilot the aircraft toward another aircraft or object. This might occur if, under conditions of high workload or stress, the human pilot might only pay attention to a limited amount of feedback and therefore not notice the feedback indicating that the modified flight path that they are proposing will pilot the aircraft toward another aircraft or object. As a result, they wrongly believe they are providing a suitable modified flight path

By studying the UCAs and scenarios generated by the updates to the STPA analysis described above, it can be observed that there are three categories of problems identified in the scenarios that cause the human pilot to issue unsafe control actions that influence the behavior of the ASEC and result in a hazard or loss. The first category is inadequate coordination between the human pilot and ASEC that can either delay the human pilot in providing the necessary confirmations or modifications to the flight path or prevent the full scope of the benefits of sharing responsibilities from being realized. As a result of inadequate coordination, the human pilot and ASEC are unable to either understand how they each generated their flight path proposals or indicate to each other what factors influenced their chosen flight path. The human pilot or ASEC therefore make independent or partially independent decisions about which flight path should be executed without fully considering all the factors identified by both of them. Two examples of these scenarios are shown in Table 44.

Table 44: Scenarios Showing Consequences of Inadequate Coordination for Part 2 Architecture Option 3

Scenario ID	Scenario
CS-G-3.1.1-2	The human pilot confirms the flight path despite receiving feedback that the flight path pilots the aircraft into another aircraft or object because although the human pilot recognizes that the flight path pilots the aircraft into another aircraft or object, they wrongly believe the other aircraft or object either is not actually present (i.e. a false positive detection) or will not actually be a collision threat in the future. As a result, the human pilot chooses to ignore the feedback indicating that the flight path proposed by the ASEC will pilot the aircraft into another aircraft or object and confirms the flight path anyway.
CS-G-1.2.1-4	The ASEC might have the correct process model of the state of the environment, the aircraft and the airspace around the aircraft and therefore plans a correct trajectory that does not pilot the aircraft toward another aircraft or object. However, when the ASEC coordinates with the human pilot, it receives an erroneous modification of the trajectory by the human pilot that does pilot the aircraft toward another aircraft or object. If the ASEC incorrectly assumes that the human-modified path is always correct, it does not validate or assess and executes it even though it flies toward another aircraft or object.

Scenarios such as these therefore show how the interface between the human pilot and ASEC must be designed to enable effective coordination between the human pilot and ASEC. For this reason, this interface between the human pilot and ASEC must not only be designed for useability and ergonomics but must also account for what the human pilot and ASEC need to receive from each other to ensure that they can effectively coordinate and make collective decisions together.

The second category is human factors issues that influence the way that the human pilot decides what inputs to provide to the ASEC. In some scenarios, human decision-making biases and heuristics might lead a human pilot to wrongly confirm or modify the ASEC's proposed flight path while in others, high workload and limited time to perform effective coordination may lead a pilot to make rushed decisions without fully considering alternatives. Examples of such scenarios are shown in Table 45.

Table 45: Scenarios Showing Consequences of Human Factors Issues for Part 2 Architecture Option 3

Scenario ID	Scenario
CS-G-3.1.1-3	Pilot confirms the flight path despite receiving feedback that the flight path pilots the aircraft into another aircraft or object. This might occur if, as a result of experience and operational adaptation, the human pilot might overly trust the ASEC to propose suitable flight paths and might become reliant on the ASEC. Such biases might therefore cause the human pilot to either assume that the ASEC is always right and confirm the flight path without checking it (i.e. essentially rubberstamping the ASEC's proposals) or perform only minimal checks to save cognitive effort. As a result, the human pilot does not notice that the flight path pilots the aircraft into another object or aircraft and confirms the flight path
CS-G-3.8.1-2	When perceiving the feedback, the human pilot may apply unsafe biases or heuristics in using that feedback to update their mental model of the state of the airspace. For example, the human pilot may only notice the most salient feedback and miss less salient feedback that would show the presence of another aircraft or object. Alternatively, feedback received by the human pilot earlier in time that does not show the presence of another aircraft or object may be weighted more heavily in their decision-making when they propose a modified flight path compared to feedback received later. As a result, they do not use the later feedback effectively to recognize that the modified flight path they are providing will pilot the aircraft toward another aircraft or object and provide a modified flight path that will pilot the aircraft toward another aircraft or object

Scenarios such as these therefore illustrate how, even though the human pilot is not in direct control over the selection of actuator movements that are provided to the aircraft, their decision-making biases and heuristics and their cognitive limitations can still lead to unsafe actuator movements being issued because this architecture option provides them with the ability to influence the ASEC's behavior. For these reasons, the design of the interface between the human pilot and ASEC must also account for these human factors issues and prevent them from leading to unsafe system behavior.

The last category is inadequate system design resulting from unsafe assumptions made by system designers. In this system, the ASEC's behavior is governed by software written by software engineers and the human pilot is at least partially dependent on the availability of (or lack of) feedback provided to them from the system. The design of the system and the software running on the ASEC as well as the assumptions that the system and software engineers make when implementing those designs must avoid unsafe assumptions that can lead to unsafe system behavior. Two example scenarios that illustrate this are shown in Table 46.

Table 46: Scenarios Showing Consequences of Unsafe Assumptions in System Design for Part 2 Architecture Option 3

Scenario ID	Scenario
CS-G-3.1.2-1	If the ASEC is unable to detect the other aircraft or object, the feedback received by the human pilot from the ASEC therefore does not show that the flight path pilots the aircraft into another aircraft or object. Furthermore, DVE conditions might prevent the human pilot from independently receiving any other feedback besides the detections made by the ASEC. As a result, the human pilot does not receive feedback that indicates that the flight path pilots the aircraft into another aircraft or object and is unable to recognize that the flight path is wrong because the human pilot can only observe what the ASEC can detect. This leads the human pilot to make the same incorrect decision that the ASEC makes, deciding that the flight path does not pilot the aircraft toward another aircraft or object.
CS-G-3.1.3-1	Although the human pilot does not confirm the flight path, a confirmation is received by the ASEC because an approval for a previously proposed flight path might be delayed in arriving. If the ASEC does not have a way to match a received approval to the flight path that the human pilot was approving, it may assume that any received approval is for the most recently proposed flight path. This asynchronicity might therefore result in approvals arriving out of order compared to the order in which flight paths are proposed. As a result, the ASEC wrongly believes the human pilot had approved the most recently proposed flight path even though the human pilot has not actually approved it yet

In some scenarios such as CS-G-3.1.2-1, if the human pilot and ASEC are only provided with the same set of feedback or if the human pilot only receives the ASEC's interpretation of sensor feedback and not the sensor feedback itself, then the human pilot can only observe what the ASEC can observe. It would therefore be much more difficult for the human pilot to recognize mistakes or inappropriate flight path proposals from the ASEC and, as a result, the benefits of sharing responsibilities may be compromised. On the other hand, in scenarios such as CS-G-3.1.3-1, assumptions encoded in the ASEC's software can lead to unsafe behavior when those assumptions turn out to be incorrect under a given set of circumstances. These scenarios therefore illustrate how important the design of the system and the assumptions that are used in the design and implementation of the system are, especially for an architecture such this where there is a heavy reliance on software-based controllers to perform safety-critical functions and where the human pilot is increasingly dependent on feedback provided to them by software-based controllers instead of having access to direct, independent feedback.

In summary, this analysis has shown that although this architecture option has the potential to reduce the workload of the human pilot by relieving them of the need to always maintain control over the aircraft under all conditions, this benefit comes with significant tradeoffs that must be carefully considered. Most critically, the interactions between the human pilot and ASEC in this architecture option can lead to new loss scenarios. In some scenarios, decision-making biases and heuristics or assumptions used by the human pilot or programmed into the ASEC software

can lead to unsafe control actions. On the other hand, in other scenarios, inadequate coordination between the human pilot and ASEC could result in unsafe behavior that may negate the benefits of sharing responsibilities if such unsafe behavior is difficult or impossible to avoid. This analysis therefore shows that although it might be beneficial for the ASEC to take over the task of flying the aircraft in real time from the human pilot, the tradeoff is that the change in architecture in this option requires the introduction of additional requirements and constraints which can increase the complexity of the system design.

4.6.8 Comparison of Architecture Options

Having described and analyzed each of the three candidate architecture options for the Piloting Controller, these three architecture options can now be compared to identify the benefits and tradeoffs of one architecture option over another that will help to inform which architecture option should be selected for further system development. For each architecture option, the benefits of that architecture option over the other 2 will be discussed and then those benefits will be contrasted with the tradeoffs or challenges that will need to be addressed if that architecture is chosen. This information together with the other analysis results described in this research should eventually be used to inform a decision when selecting one architecture option for further system development.

As described in the introduction to the architecture options, the system architecture in option 1 relies the least on automation to assist in the tasks necessary for safe flight in DVE and therefore most of the responsibilities that are identified are assigned to the human pilot. The ASEC thus serves simply as a decision aid, assisting the human pilot by providing them with feedback from the aircraft sensors that the human pilot can use to make decisions when selecting actuator movements to fly the aircraft. For these reasons, compared to options 2 and 3, the benefits of architecture option 1 are that there are only a few responsibilities that are shared between the human pilot and ASEC and therefore less coordination is required between the human pilot and ASEC to avoid unsafe control actions. Consequently, there are few, if any, scenarios involving inadequate coordination between the human pilot and ASEC that need to be mitigated or prevented. Compared to architecture options 2 and 3, this reduces the number of constraints that are needed to avoid unsafe system behavior. Furthermore, because the ASEC is only assigned relatively simple responsibilities to carry out, there are also fewer challenges related to the design and implementation of the ASEC that need to be addressed to ensure that the ASEC's software design does not contribute to unsafe system behavior. For these reasons, option 1 has the potential to be a simpler and less complex system design compared to options 2 and 3.

However, because option 1 relies primarily on the human pilot to carry out many of the responsibilities, several human factors challenges will need to be addressed to ensure that the architecture supports the responsibilities assigned to the human pilot and does not instead make it harder for the human pilot to carry out their assigned responsibilities. As discussed in the architecture analysis for option 1 in Section 4.6.3, two main challenges for this architecture option were identified:

1. Feedback mechanisms must be designed to help the human pilot avoid relying on unsafe human decision-making biases and heuristics
2. The human pilot will experience an increased workload when carrying out the responsibilities necessary to ensure safe flight in DVEs. This increased workload results from three main factors:
    a. Increased cognitive effort required to validate feedback and update their mental model
    b. Difficult conditions under which the human pilot is expected to make decisions. These conditions include the need to make decisions quickly and accurately as well as the need to make decisions under uncertainty or with incomplete information
    c. The larger mental model that the human pilot will need to maintain

For these reasons, if architecture option 1 is selected, it will be important to account for human factors considerations when designing the feedback mechanisms as well as the displays and other pilot interfaces to ensure that the human pilot is able to effectively carry out the responsibilities assigned to them and avoid making the already high anticipated workload even higher.

One way to alleviate the potentially high workload imposed on the human pilot by architecture option 1 could be to reduce the size of the mental model that the human pilot needs to maintain so that the human pilot has fewer process model parts to maintain in memory or keep updated. Unfortunately, the three architecture options show how difficult it is to reduce the size of the mental model that the human pilot must maintain. Although architecture option 2 assigns more responsibilities to the ASEC, the number of parts of the process model that are assigned to the human pilot does not change. It is only in architecture option 3 when the human pilot is no longer assigned responsibilities for directly controlling the aircraft and the ASEC is assigned control over the aircraft that the mental model of current weather conditions no longer needs to be assigned to the human pilot. This thus shows how difficult it is to reduce the size of the human pilot's mental model because the ASEC had to advance from being a decision aid in option 1 to being in direct control over the aircraft to reduce the size of the human pilot's mental model by 1 part.

An alternative way to reduce the workload of the human pilot is to relieve the human pilot from carrying out some responsibilities or provide them with assistance in carrying out their assigned responsibilities. Architecture option 2 therefore does this by sharing some of the responsibilities with the ASEC so that the ASEC can assist by proposing flight paths or actuator movements for the human pilot's consideration. In doing so, architecture option 2 offers several benefits. Compared to architecture option 1, option 2 not only offers the potential to reduce the workload of the human pilot but it also provides the human pilot with assistance in interpreting feedback or considering various alternatives to support the human pilot's decision making, especially under cognitively challenging circumstances or when decisions need to be made under uncertainty or with incomplete information. In addition, compared to architecture option 3, option 2 keeps the human pilot involved in selecting actuator movements. As a result, it is easier for the human pilot to modify or override the ASEC's proposed flight paths because the human pilot is primarily in control over the aircraft.

However, these benefits have associated tradeoffs. As discussed in the architecture analysis for option 2, the sharing of responsibilities result in two challenges. The first challenge is that not only do the feedback mechanisms still need to be designed to help the human pilot avoid relying on unsafe human decision-making biases and heuristics, the ASEC's software must also avoid having human decision-making biases programmed into it to ensure that the benefits of sharing responsibilities between the human pilot and ASEC are achieved. If the ASEC's software has similar (or worse) decision-making biases or heuristics programmed into it, then its behavior may resemble those of the human pilot, negating some of the benefits of shared responsibilities. The second challenge is that the sharing of responsibilities can increase the complexity of the system because additional requirements are needed to ensure adequate coordination is achieved between the human pilot and ASEC. If there is inadequate coordination, additional loss scenarios may occur that are more difficult to mitigate or prevent. In addition, the benefits of sharing responsibilities may be negated or the workload imposed on the human pilot may inadvertently be increased. As such, the benefits of this architecture option compared to the other two should be compared against these challenges. If this architecture option is chosen, not only do the human factors considerations and requirements on feedback design need to be accounted for, additional constraints and requirements will need to be imposed on the system to prevent unsafe system behavior caused by inadequate coordination between the human pilot and ASEC.

While architecture option 2 provides the human pilot with assistance in carrying out responsibilities where increased cognitive effort is required or when complex decisions need to be made, that option does not assist the human pilot in carrying out responsibilities such as SR-21 and SR-22 that require decisions to be made quickly and accurately. As such, architecture option 3 addresses this challenge by transferring those two responsibilities to the ASEC and applying the same assignments as architecture option 2 for the remaining responsibilities and the process model parts. As a result, compared to option 2, architecture option 3 attempts to address most of the challenges identified in the architecture analysis of option 1 by leveraging the most use of automation out of all three architecture options. This allows the ASEC to not only help the human pilot detect weather conditions and other aircraft and objects but to also control the aircraft. The system is therefore no longer dependent on the human pilot providing actuator movements to the aircraft subsystems quickly and accurately.

However, as might be expected, this architecture option not only inherits all the tradeoffs identified for option 2 but also incurs some additional tradeoffs due to the change in the role of the human pilot from a system operator in architecture option 1 and 2 to a supervisor in architecture option 3. As discussed in the architecture analysis for option 3, these additional tradeoffs arise due to the additional opportunities for inadequate coordination between the human pilot and ASEC or inadequately defined system or software requirements that lead to unsafe system behavior. Because the ASEC is the most heavily involved in flying the aircraft in this option compared to options 1 and 2, it is even more important that the interfaces between the human pilot and the ASEC be designed to enable effective coordination between the human pilot and ASEC. This ensures that they can communicate with each other and perform effective group decision-making. In addition, although the human pilot is no longer directly involved in flying the aircraft in this architecture option, unsafe human decision making biases and heuristics

can still lead to unsafe system behavior because the human pilot can influence the behavior of the ASEC. For these reasons, the additional requirements and constraints needed to avoid these causal factors from leading to a hazard or loss may make the system more challenging to design and may also increase the complexity of the system.

In summary, comparing the three architecture options, the level of automation and the level of involvement of the ASEC in the task of flying the aircraft increases from option 1 to option 3 and this increase in the use of automation has the potential to reduce the workload of the human pilot and provide them with assistance in flying the aircraft safely in DVE. However, as the level of automation increases, so too does the system complexity and the challenges in being able to successfully design and implement such a system architecture. As the comparison of architecture options has shown, not only does the design of the feedback mechanisms and interactions between the human pilot and the system need to account for human factors considerations but there are also an increasing number of ways that inadequate coordination between the human pilot and ASEC or unsafe assumptions encoded into the system software can lead to unsafe system behavior as the level of automation increases. As such, the benefits of increasing the system's reliance on automation must be balanced against the tradeoffs in this architecture analysis to determine if the benefits gained are worth the tradeoffs incurred. Eventually, the information discussed in this analysis should be used to inform a decision when selecting one of these architecture options to proceed with further system development.

Chapter 5 Conclusions

5.1 Summary

This thesis developed a new approach to architecture development that overcomes limitations of current methods, can be applied early in the design process and is appropriate for today's increasingly complex systems. Using the new approach, a system can be analyzed in the earliest stages of system development to create system architectures in a top-down and safety-driven manner.

Unlike current methods for architecture development that rely on decomposition to create system architectures, this new approach uses appropriate types of abstraction to guide the design process and organize the design information. It begins by analyzing the system in its environment using STPA to identify the system-level interactions that could lead to unsafe behaviors and then uses the analysis results to drive the identification of solution-neutral, system-level requirements. This ensures that safety or other emergent properties are designed into the system from the beginning. Once these system-level requirements have been identified, they can be used to generate the system-level behavior that describes how the system must behave to fulfill the system requirements. Finally, the system-level behavior information is used to inform the creation and assessment of architecture options for the overall system as well as each of the components. By applying this new approach iteratively, a conceptual architecture for a system can be created and refined until the detailed design for a system is complete.

There are four key features that differentiate this new approach from current methods for architecture development:

1. It applies a systems-theoretic approach by proceeding top-down and considering the system as a whole and the interactions between system components instead of analyzing the components individually
2. It integrates STPA as the hazard analysis early in the design process. This ensures that emergent properties such as safety are designed into the system from the beginning and enables considerations from other disciplines such as human factors to be integrated into the analysis.
3. It assists system designers in obtaining more information about what is needed to carry out the responsibilities effectively before architecture options are created. This enables architectures to be designed to ensure that the responsibilities can be carried out as effectively as possible.
4. It uses means-ends abstraction to guide the design process instead of being reliant on functional or physical decomposition to create system architectures. This helps system designers and reviewers to manage system complexity and identify what the architecture must do and how it must behave before architectural decisions are made

This thesis also demonstrated this new approach by applying it to create the architecture for a system involving a human pilot and automated aircraft controller flying aircraft in DVEs. Using

the new approach, the Flight Operations System needed to safely perform medevac flights in DVEs was analyzed using STPA to generate a set of system-level requirements. The system-level requirements were then used to generate the system-level behavior information that described the responsibilities necessary for safe flight as well as the process model parts, feedback sources and timing requirements that were necessary to enable safe flight. This system-level behavior information was then used to inform the creation of candidate architectures. First, two architecture options for the overall Flight Operations System were analyzed and the tradeoffs between the architecture options was compared to inform the selection of one option for further development. The chosen architecture was then used to develop and analyze architecture options for the human pilot and aircraft software controller. The creation and assessment of these architecture options thus demonstrated how the system design information generated using the method along with further STPA and other analyses can be used to identify the tradeoffs between architecture options to inform a decision about which architecture should be selected for further system development.

5.2 Future Work

Although this new approach to concept and architecture development addresses the three challenges for creating and assessing system architectures, there are several ways that this new approach could be further refined to provide additional assistance to system designers and reviewers in understanding the system to design or reviewing it.

The first is that the decision-making strategy could be defined more formally to avoid ambiguity and make it easier to translate it into other forms more suitable for detailed system design and implementation. In this research, the decision-making strategy is defined in a relatively unstructured manner using natural language. As a result, there can be ambiguities in interpreting the decision-making strategy or using it create other design artifacts such as a black-box specification for software development. For these reasons, it may be beneficial to define the decision-making strategy more formally to reduce ambiguity. However, because the decision-making strategy is a part of the system-level behavior information, it must still be presented in a readable format that is easy for anyone who might use the system the specification to understand. As such, further research is needed to determine if a specification language such as SpecTRM-RL [31] that is both formal and readable could be used instead of natural language for defining the decision-making strategy.

The second is that the creation and assessment of architecture options requires additional structure and guidance to ensure a more systematic approach to creating and assessing these options. In this research, the architecture options were created and assessed by essentially applying known or familiar heuristics that are usually used when defining these types of architectures for aircraft. However, to create truly novel or new designs, the process of creating and assessing architectures must assist system designers in thinking beyond known or familiar heuristics or reference designs to help them identify new types of architectures based on the specific problem they are trying to solve. As such, further research is needed to determine how the architecture creation and assessment could be better structured to help system designers

consider other types of architectures instead of relying on familiar heuristics or reference designs.

The third is that the creation and assessment of architecture options also needs to incorporate information from upstream influences in the system design process. In this research, architecture options were created and assessed based only on the requirements and system-level behavior information generated using this new approach along with human factors and other considerations. However, upstream influences such as stakeholder preferences and priorities should also influence the system design such that different stakeholder preferences and priorities result in different architectures even when the requirements and system-level behavior information remain constant. As such, further research is needed to determine how to incorporate information from upstream influences such as a stakeholder analysis when creating and assessing architecture options.

Chapter 6 References

[1] "Systems Engineering for Intelligent Transportation Systems: An Introduction for Transportation Professionals." Department of Transportation, Office of Operations, Jan. 2007.

[2] 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.

[3] N. Leveson, "An Improved Design Process for Complex Control-Based Systems Using STPA and a Conceptual Architecture," Massachusetts Institute of Technology, White Paper, 2019.

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

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

[6] "ISO/IEC/IEEE 42010 Systems and software engineering - Architecture description," IEEE. doi: 10.1109/IEEESTD.2011.6129467.

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

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

[9] "ISO/IEC/IEEE 15288 Systems and software engineering - System Lifecycle Processes," IEEE.

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

[11] J. Rasmussen, "The role of hierarchical knowledge representation in decision making and system management," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. SMC-15, no. 2, pp. 234–243, Mar. 1985.

[12] 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.

[13] "Unmanned Aircraft Systems (UAS) Traffic Management (UTM) Concept of Operations v2.0," Federal Aviation Administration, Department of Transportation, Mar. 2020.

[14] N. G. Leveson and J. P. Thomas, *STPA Handbook*. 2018.

[15] K. H. Johnson, "Systems-Theoretic Safety Analyses Extended for Coordination," MIT Department of Aeronautics and Astronautics, PhD Dissertation, Feb. 2017.

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

[17] H. M. Slominski, "Using STPA and CAST to Design for Serviceability and Diagnostics," MIT System Design and Management Program, Masters Thesis, May 2020.

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

[19] N. G. Leveson, "Rasmussen's legacy: A paradigm change in engineering for safety," *Applied Ergonomics*, vol. 59, pp. 581–591, Mar. 2017.

[20] N. G. Leveson, “The drawbacks in using the term ‘system of systems,’” *Biomedical Instrumentation & Technology*, vol. 47, no. 2, pp. 115–118, 2013.
[21] “Spatial Disorientation Induced by a Degraded Visual Environments: Training and Decision-Making Solutions Recommended Practice,” United States Helicopter Safety Team, Dec. 2020.
[22] S. P. Baker, J. G. Grabowski, R. S. Dodd, D. F. Shanahan, M. W. Lamb, and G. H. Li, “EMS Helicopter Crashes: What Influences Fatal Outcome?,” *Annals of Emergency Medicine*, vol. 47, no. 4, pp. 351–356, Apr. 2006.
[23] J. Vreeken, “Helicopter Flight in a Degraded Visual Environment.” National Aerospace Laboratory, 2013.
[24] J. Keller, “Three companies to develop synthetic-vision avionics to help land helicopters in choking dust,” *Military & Aerospace Electronics*, Oct. 03, 2013. [Online]. Available: https://www.militaryaerospace.com/computers/article/16715788/three-companies-to-develop-syntheticvision-avionics-to-help-land-helicopters-in-choking-dust
[25] “Emergency Medical Services Policies and Procedures.” County of Ventura Health Care Agency, Dec. 01, 2011.
[26] “EMS Aircraft Dispatch-Response-Utilization Policies and Procedures.” Kern County Public Health Services Department, Apr. 11, 2002.
[27] “Air Medical Services (AMS) Guidelines for all first responders and EMS agencies serving the Hudson Valley & Westchester EMS Regions.” Hudson Valley-Westchester Inter-Regional Helicopter Committee, Jan. 2021.
[28] “AC 135-14B: Helicopter Air Ambulance Operations.” Federal Aviation Administration, Mar. 26, 2015.
[29] R. Cabosky, “Application of Hierarchy to STPA: A Human Factors Study on Vehicle Automation,” MIT Department of Aeronautics and Astronautics, Masters Thesis, Aug. 2020.
[30] C. D. Wickens, Ed., *An introduction to human factors engineering*, 2. ed. Upper Saddle River, NJ: Pearson/Prentice Hall, 2004.
[31] N. Leveson, “An Introductory Guide to SpecTRM.” Safeware Engineering Corp, 2005. [Online]. Available: https://dspace.mit.edu/bitstream/handle/1721.1/71860/16-358j-spring-2005/contents/assignments/tutorial.pdf

Appendix A UCAs Identified For STPA Analysis of FOS

As part of this research, UCAs were identified for the Piloting Controller (Table 47) to demonstrate how the control structure shown in Figure 16 should be analyzed. Although UCAs for other controllers were not analyzed as part of this research project, a complete STPA analysis of the control structure should identify UCAs for all control actions in the control structure.

Table 47: UCAs for the Piloting Controller

Control Action	Providing	Not Providing	Too Early/Too Late	Applied Too Long/Stopped Too Soon
Actuator Movements	UCA-1.1: Piloting Controller provides actuator movements during takeoff when the aircraft is not in a safe departure state [H-3, H-4, H-5] UCA-1.2: Piloting Controller provides actuator movements that steers the aircraft toward another aircraft or object [H-3] UCA-1.3: Piloting Controller provides actuator movements when the maneuver will	UCA-1.5: Piloting Controller does not provide actuator movements when violation of minimum separation is imminent [H-3] UCA-1.6: Piloting Controller does not provide actuator movements during critical phases of flight [H-1, H-3]	UCA-1.7: Piloting Controller provides actuator movements too late during takeoff after takeoff clearance is granted when another aircraft or obstacle has entered the airspace near the aircraft [H-3] UCA-1.8: Piloting Controller provides actuator movements too late when the aircraft is near another aircraft or obstacle [H-2, H-3] UCA-1.9: Piloting Controller provides actuator movements too early during landing before the aircraft has cleared all vertical	UCA-1.10: Piloting Controller applies actuator movements for too long during takeoff after it has passed the desired altitude [H-1, H-3] UCA-1.11: Piloting Controller stops providing actuator movements too soon during takeoff before the desired altitude is reached [H-3] UCA-1.12: Piloting Controller stops applying actuator movements too soon before the aircraft has

	place the airframe under extreme stresses [H-1, H-2] UCA-1.4: Pilot provides actuator movements when the aircraft has already landed [H-1, H-2, H-4]		obstacles on the ground [H-3]	landed [H-2, H-4]

Appendix B Loss Scenarios Identified For STPA Analysis of FOS

Scenarios for UCA-1.1

UCA-1.1: Piloting Controller provides actuator movements during takeoff when the aircraft is not in a safe departure state [H-3, H-4, H-5]

CS-1.1.1: The piloting controller provides actuator movements during takeoff despite receiving feedback that the aircraft is not in a safe departure state. This could occur because:

1. The piloting controller has the wrong process model of the state of the aircraft and therefore believes that the aircraft is in a safe departure state. This could occur if:
   1.1. The piloting controller receives the feedback indicating that the aircraft is not in a safe departure state but does not incorporate that feedback into its process model of the state of the aircraft because:
      1.1.1. The piloting controller may wrongly believe that the data is too old/out-of-date. As a result, the piloting controller ignores that feedback, believing that it should make use of newer data to update its process model of the state of the aircraft. [SLR-1]
      1.1.2. The piloting controller believes it is receiving conflicting feedback because DVE conditions have degraded the accuracy of or obscured some (but not all) feedback sources. The piloting controller therefore chooses to ignore the feedback showing the presence of the other aircraft or object, believing that it is a false positive or erroneous. The piloting controller is especially susceptible to this if the majority of sources appear to agree that no aircraft is present and fewer sources appear to show otherwise. As a result, the piloting controller updates its process model based only on what it wrongly believes to be the correct feedback. [SLR-2]
      1.1.3. Alternatively, if the piloting controller receives communication from another controller (e.g. another aircraft, ground personnel, EMS Operations and ATC) indicating that the aircraft is in a safe departure state, the piloting controller may assume that the communication from the other controller must be accurate. This may even be used as proof that the feedback showing that the aircraft is not in a safe departure state should be ignored as out-of-date or incorrect. As a result, the piloting controller uses the communication from the other controller to update its process model and therefore wrongly believes that the aircraft is in a safe departure state. [SLR-3]
   1.2. Under the time constraints of needing to depart for a mission, the piloting controller either decides to or is told by another controller to expedite departure checks. This could be the result of unsafe operational adaptation to speed up the time required to prepare the aircraft for departure or the result of a capability to override or otherwise influence the checks. As a result, the piloting controller does not perform a complete update of its process model of the state of the aircraft and therefore might believe that, based on what it did check, the aircraft is in a safe departure state. [SLR-4]
   1.3. The piloting controller had the correct process model of the state of the aircraft and the expected safe departure state at the point that the aircraft was checked. However,

prior to departure, the state of the aircraft changes and the piloting controller does not update the process model, believing that there is no need to check the state of the aircraft again. As a result, they are unaware that the aircraft is no longer in a safe departure state [SLR-5]

2. The piloting controller has a wrong/incomplete process model of the expected safe departure state of the aircraft and therefore decides based on this wrong/incomplete model that the current state of the aircraft matches the desired safe departure state. This could occur if:
   2.1. The data (e.g. aircraft manifest) provided to the piloting controller (e.g. from EMS Operations and ATC or ground personnel) indicating the expected safe departure state of the aircraft was incompletely specified but is assumed by the piloting controller to be complete. The piloting controller therefore updates its process model of the desired safe departure state of the aircraft based on this wrong or incomplete information. [SLR-6]
   2.2. A last-minute update is made to the expected safe departure state of the aircraft and the piloting controller does not receive the update. This could occur if:
      2.2.1. The update was not provided to the piloting controller due to poor communication [SLR-7*]*
      2.2.2. Any of the conditions in CS-1.1.1-1.1 above may occur, resulting in the piloting controller receiving the update but not incorporating it into its process model of the desired safe departure state. [SLR-1, SLR-2, SLR-3]

CS-1.1.2: The pilot receives feedback that did not indicate that the aircraft was not in a safe departure state because:

1. The feedback indicating that the aircraft was in an unsafe departure state was available from the feedback mechanisms/sensors (e.g. reporting systems, aircraft maintenance logs, warnings/alarms from the aircraft subsystems). However, that data is not received by the piloting controller. If the piloting controller assumes/is programmed to assume that not receiving such feedback indicates that there are no outstanding maintenance or configuration updates, then the piloting controller will make an incorrect update of its process model of the state of the aircraft and believe the aircraft is in a safe departure state. Data might be collected by the sensors/feedback mechanisms but not be received by the piloting controller because:
   1.1. The data is missed/dropped before it can be processed by the piloting controller. This could occur because the piloting controller was preoccupied and unable to process the incoming data or the piloting controller was unaware of the arrival of the data to be processed [SLR-8]
   1.2. The required data is not made available to the piloting controller even though it is collected. Examples of this include warnings from the aircraft subsystems that are buried deep in menus on the display interfaces (if the piloting controller is human), data not being distributed to the automation (if the piloting controller is automated) or the piloting controller not having access to manifests, maintenance logs or the status of preflight activities such as configuration updates. [SLR-9]

1.3. Alternatively, the data is made available but the piloting controller does not check those data collection mechanisms (e.g. maintenance reports, warning messages) prior to departure [SLR-9]
1.4. For maintenance or configuration tasks, an update could have been made (or there was a delay in updating records) showing an outstanding task or issue after the piloting controller has checked the data collection mechanisms. If the piloting controller is not notified of this update, then the piloting controller does not have another opportunity to update their process model of the state of the aircraft [SLR-10]

2. The sensor data required to indicate that the aircraft was in an unsafe departure state is not available. As a result, no data is collected regarding those aspects of a safe departure state and the piloting controller is unaware that those aspects are in an unsafe state. The sensor data might not be available because:
   2.1. Maintenance or preflight records for the aircraft are not updated to show the incomplete or outstanding maintenance task(s) or the state of each task. This could occur due to delays in updating those records after an issue is discovered or a technical issue preventing the update of those records [SLR-10]
   2.2. Sensors used to detect the unsafe departure state of the aircraft have failed and their failure does not trigger a warning of the failure from the aircraft subsystems [SLR-9]
   2.3. DVE conditions obscure the area around the aircraft and prevents the piloting controller from seeing if any maintenance or preflight activities are still being performed on the aircraft (either visually or via other sensor modalities). [SLR-3, SLR-9]

CS-1.1.3: The piloting controller does not provide actuator movements during takeoff but actuator movements are received by the aircraft because:

1. An adversary spoofs the actuator movements sent from the piloting controller and causes the aircraft to believe the piloting controller has sent actuator movements [SLR-11]

CS-1.1.4: Actuator movements are not received by the aircraft but the aircraft behaves as though actuator movements have been received and violates minimum separation or is harmful to human health because:

1. A subsystem controller fails, sending actuator movements when none were commanded by the piloting controller [SLR-12]

Scenarios for UCA-1.2

UCA-1.2: Piloting controller provides actuator movements that pilots the aircraft toward another aircraft or object [H-3]

CS-1.2.1: The piloting controller provides actuator movements despite receiving feedback that there was another aircraft or object nearby. This could occur because:

1. The piloting controller may have the wrong process model of the state of the environment, the aircraft or the airspace around the aircraft. As a result, the piloting controller might wrongly believe that the airspace nearby the aircraft is clear to pilot toward or select actuator movements that, because of the wrong process model of the state of the aircraft

or environmental conditions, cause the aircraft to move toward instead of away from the aircraft or object. This may occur because:

1.1. DVE conditions cause more noise in sensor data than is normally present, making the useful feedback more difficult for the piloting controller to distinguish from the noise or increasing the likelihood that the piloting controller wrongly decides that the sensor data shows no useful feedback/detections. [SLR-13, SLR-19]

1.2. Any of the conditions in CS-1.1.1-1.1 could occur in this case as well [SLR-1, SLR-2, SLR-3]

1.3. DVE causes delays in the piloting controller interpreting the sensor data and using the sensor data to update its process model. As a result, the piloting controller has the wrong process model of the airspace around the aircraft until it is able to update its process model. [SLR-14]

1.4. If another aircraft or controller external to the aircraft providing airspace guidance is in contact with the aircraft and wrongly believes that the airspace nearby the aircraft is clear (for any of the same reasons above) or if a-priori data indicates that no ground obstacles are present, the piloting controller may assume that the external controller/a-priori data is providing accurate information and disregard the onboard sensors in favor of the guidance provided by the external controller. If the piloting controller is human, this may also contribute to confirmation bias, causing the piloting controller not to look for any other evidence that another aircraft or object is nearby. As a result, the piloting controller wrongly updates its process model of the airspace nearby the aircraft [SLR-3, SLR-15]

1.5. The piloting controller had the correct process model of the airspace at the point that it checked. However, between the time that the piloting controller checked the state of the airspace and the time that it begins its maneuver, an object or other aircraft enters the nearby airspace. However, the piloting controller, believing that the airspace is clear, does not check that airspace again and therefore now has the wrong process model of the nearby airspace when it actually begins its maneuver. [SLR-16]

2. The piloting controller may have the wrong belief about the future behavior of the other aircraft or object and therefore believes that providing actuator movements will not pilot the aircraft toward the other aircraft or object. This might occur because:

2.1. The other aircraft does not coordinate its movements sufficiently with this aircraft and therefore the piloting controller is forced to make assumptions based on the movement patterns it has observed and predict the other aircraft's intent [SLR-17, SLR-18]

2.2. The other aircraft does coordinate its movements with this aircraft but, at the last minute or for some other reason, is forced to change its intended movements but does not communicate this new intent to the piloting controller [SLR-17, SLR-18]

3. The piloting controller is forced to make a quick decision to avoid violating minimum separation that does not fully account for all objects and aircraft in the airspace. As a result, the piloting controller tries to avoid one object/aircraft and collides with another aircraft/object instead. [SLR-17, SLR-18]

CS-1.2.2: The piloting controller receives feedback that did not indicate that another aircraft or object was nearby because:

1. DVE degrades or obscures sensors, leading to wrong, incomplete or missing feedback about the environment, DVE conditions or the state of the aircraft. This could also be caused by a sensor suite that was not designed to operate in a particular set of DVE conditions [SLR-2, SLR-15, SLR-19]
2. Changes in aircraft position are so slow/subtle that, especially under some DVE conditions, the piloting controller is unable to distinguish the feedback showing the aircraft shifting or drifting in position. As a result, the pilot's mental model of the state or position of the aircraft does not match the aircraft's actual state or position [SLR-20]
3. The feedback indicating the presence of another aircraft or object nearby was delayed in arriving. This might occur because DVE conditions increase the processing time needed before the piloting controller receives the feedback. As a result, the piloting controller selects actuator movements based on its current process model of the airspace which does not reflect the presence of another aircraft or object. [SLR-14, SLR-21]
4. Feedback not showing the presence of another aircraft or object nearby was delayed in arriving. As a result, if the piloting controller is not aware of the delay, it might wrongly choose to incorporate that feedback even though it is already out-of-date. [SLR-1]

CS-1.2.3: The piloting controller does not provide actuator movements that pilots the aircraft toward another aircraft or object but actuator movements to do so are received by the aircraft because:

1. A similar cybersecurity reason as CS-1.1.3-1 would apply here as well [SLR-11]

CS-1.2.4: Actuator movements are not received by the aircraft but the aircraft still violates minimum separation with the nearby aircraft or object because:

1. DVE conditions (e.g. wind gusts) move the aircraft toward the other aircraft or object with enough force or at a high enough rate that the piloting controller is unable to react quickly enough or with appropriate amplitude to correct the disturbance [SLR-22]

Scenarios for UCA-1.5

UCA-1.5: Piloting Controller does not provide actuator movements when violation of minimum separation is imminent [H-3]

CS-1.5.1: The piloting controller does not provide actuator movements despite receiving feedback that violation of minimum separation is imminent. This could occur because:

1. The piloting controller may be unable to select new actuator movements before the violation of minimum separation occurs. This could occur if:
   1.1. The piloting controller is unable to select new actuator movements quickly enough to avoid violation of minimum separation. This might occur if the piloting controller recognizes the imminent violation too late or takes too long to select new actuator movements [SLR-23]

1.2. There is no viable path that would allow the piloting controller to avoid the current violation of minimum separation without causing another one (i.e. no viable escape path) [SLR-24]

2. The piloting controller has the wrong belief about the future movement of the other aircraft or object and believes that the other aircraft or object will no longer pose a threat. [SLR-18]

CS-1.5.2: The piloting controller receives feedback that did not indicate that violation of minimum separation is imminent. These scenarios are the same as CS-1.2.2.

CS-1.5.3: The piloting controller does provide actuator movements but actuator movements are not received by the aircraft because:

1. A similar cybersecurity reason as CS-1.1.3-1 would apply here as well [SLR-11]

CS-1.5.4: Actuator movements are received by the aircraft but the aircraft still violates minimum separation because:

1. The piloting controller may have the wrong process model of the environment conditions and selects actuator movements that are insufficient to effect the desired change in flight path [SLR-25]
2. The piloting controller selects actuator movements that avoid one violation of minimum separation but causes another one [SLR-25, SLR-26]

Appendix C System-Level Requirements

Table 48 shows how system-level requirements should be recorded along with traceability links to the scenarios they are meant to mitigate or prevent as well as the underlying rationale and assumptions.

Table 48: System-Level Requirements

Req ID	System-Level Requirement	Scenario Links	Rationale/Assumptions
SLR-1	The FOS must be able to determine if any feedback it is receiving about the aircraft's mission readiness, state of the aircraft and the airspace around it is too old	CS-1.1.1-1.1.1, CS-1.2.1-1.2, CS-1.2.2-4	Assumes that it is possible to determine the validity/expiry period for all feedback information
SLR-2	The FOS must account for prevailing DVE conditions and their possible effect on feedback sources when making use of feedback information	CS-1.1.1-1.1.2, CS-1.2.1-1.2, CS-1.2.2-1	Assumes that there is minimal ambiguity in deconflicting feedback that appears to be in conflict once a framework/guidance is established for determining how feedback sources might be affected by DVEs
SLR-3	The FOS must verify the accuracy of any inputs from another aircraft/controller before updating process models based on that input	CS-1.1.1-1.1.3, CC-1.1.2-2.2.2, CS-1.2.1-1.2, CS-1.2.1-1.4	Assumes that all possible inputs from external sources can be verified in some manner using other available data
SLR-4	The FOS must confirm that all aspects of the aircraft state match the safe departure state prior to departure	CS-1.1.1-1.2	
SLR-5	The FOS must always be able to determine if the state of the aircraft changes between the time that it was checked and departure	CS-1.1.1-1.3	

Req ID	System-Level Requirement	Scenario Links	Rationale/Assumptions
SLR-6	The FOS must confirm that data provided to it about the expected safe departure state of the aircraft has been fully specified	CS-1.1.1-2.1	Assumes that it is possible to unambiguously define when the safe departure state is “fully specified” and evaluate whether that has been achieved
SLR-7	The FOS must ensure that all relevant personnel are aware of any changes to the expected safe departure state of the aircraft	CS-1.1.1-2.2.1	
SLR-8	The FOS must process all feedback to make a deliberate decision if it is to be ignored/dropped	CS-1.1.2-1.1	
SLR-9	The FOS must ensure that it is receiving all data required to determine the expected safe departure state and determine the current state of the aircraft under all DVE conditions at all times	CS-1.1.2-1.2, CS-1.1.2-1.3, CS-1.1.2-2.3	
SLR-10	The FOS must ensure that flight crews are aware of the most updated state of any maintenance tasks on the aircraft	CS-1.1.2-1.4	
SLR-11	The FOS must ensure that only authorized actuator movements are executed	CS-1.1.3-1, CS-1.5.3-1	
SLR-12	All FOS equipment and systems must be able to operate in the expected DVE conditions	CS-1.1.4-1	Assumes that component failure can be avoided if components are designed to operate in expected DVE conditions
SLR-13	The FOS must be able to distinguish useful detections/feedback from the noise that might be present in feedback data under all DVE conditions at all times	CS-1.2.1-1.1	

Req ID	System-Level Requirement	Scenario Links	Rationale/Assumptions
SLR-14	The FOS must be able to process sensor data and use it to update its process model sufficiently quickly (within TBD seconds)	CS-1.2.1-1.3, CS-1.2.2-3	<Rationale for the threshold time in which sensor data must be processed>
SLR-15	The FOS must receive all data required to determine the state of the environment and conditions around the aircraft under all DVE conditions at all times	CS-1.2.1-1.4, CS-1.2.2-1	
SLR-16	The FOS must always be able to determine if the state of the airspace changes between the time that it was checked and the commencement of a maneuver	CS-1.2.1-1.5	
SLR-17	The FOS must take into account the current and future movements of other aircraft in the vicinity when selecting actuator movements	CS-1.2.1-2.1, CS-1.2.1-2.2, CS-1.2.1-3	Assumes that even without the other aircraft coordinating sufficiently, there is still enough information available to select actuator movements that avoid violation of minimum separation
SLR-18	The FOS must ensure that aircraft movements are selected such that sufficient reaction time is available to react and prevent violation of minimum separation if intent or movements of the other aircraft are different than expected	CS-1.2.1-2.1, CS-1.2.1-2.1, CS-1.2.1-3, CS-1.5.1-2	Assumes that there is enough clear airspace nearby to leave enough space between aircraft such that enough reaction time is available if an aircraft moves in an unexpected way
SLR-19	The FOS must be able to detect all objects and other aircraft in the environment under all DVE conditions at all times	CS-1.2.1-1.1, CS-1.2.2-1	
SLR-20	The FOS must be able to detect slow or subtle changes in the state of the aircraft under all DVE conditions at all times	CS-1.2.2-2	

Req ID	System-Level Requirement	Scenario Links	Rationale/Assumptions
SLR-21	The FOS must select actuator movements that minimize the risk of violation of minimum separation when information about the state of the airspace is in a degraded condition (e.g. delayed)	CS-1.2.2-3	Assumes that it is possible to find a set of actuator movements that presents the lowest risk of violating minimum separation. This requires both a framework for making that evaluation and enough space between aircraft that a relatively low-risk solution can be found
SLR-22	The FOS must be able to respond quickly enough and with an appropriate magnitude to disturbances to prevent unintended movements of the aircraft	CS-1.2.4-1	Assumes that there are no conditions under which unintended movements of the aircraft is unavoidable. Also assumes that it is desirable to always avoid undesirable movement of the aircraft
SLR-23	The FOS must be able to respond quickly enough to avoid violations of minimum separation	CS-1.5.1-1.1	
SLR-24	The FOS must ensure that a viable path is always available to avoid a violation of minimum separation	CS-1.5.1-1.2	Assumes that there is enough space between aircraft to be able to find a viable path to avoid violating minimum separation
SLR-25	The FOS must select actuator movements that are sufficient and appropriate to effect the desired change in flight path under the given environmental conditions	CS-1.5.4-1, CS-1.5.4-2	

Req ID	System-Level Requirement	Scenario Links	Rationale/Assumptions
SLR-26	The FOS must ensure that actuator movements avoid all possible violations of minimum separation	CS-1.5.4-2	Assumes that it is possible to avoid all violations of minimum separation. This includes not only enough space between aircraft but also that a framework exists for how to avoid multiple possible violations at once

Appendix D FOS System-Level Behavior Information

Decision-Making Strategy, Refined Control Actions and Required Process Model Parts

Table 49: Full Details for Decision-Making Strategy, Refined Control Actions and Required Process Model Parts

Resp. ID	Responsibility	Decision-Making Strategy	Required Process Model Contents	Rationale and Assumptions
SR-1.	Determine if feedback received about the aircraft's mission readiness, state of the aircraft and the airspace is too old [SLR-1]	FOS will compare the timestamp of the feedback with timestamp for the current time and calculate the difference. If the difference exceeds <TBD threshold>, the feedback is considered too old	Model of System Behavior • Threshold for out-of-date data Aircraft Mission Readiness, State of the aircraft, Current state of the airspace • Time of feedback measurement	<Rationale for how that threshold is determined> <Assumptions with respect to always having that timestamp available> Assumes that the threshold is constant in all DVE and operational conditions (e.g. will not change based on speed of flight)

Resp. ID	Responsibility	Decision-Making Strategy	Required Process Model Contents	Rationale and Assumptions
SR-2.	Account for prevailing DVE conditions and their possible effect on feedback sources when making use of feedback information [SLR-2]	FOS will compare current weather conditions to its understanding of which data sources are affected by which DVE conditions to determine which feedback sources to trust	Model of system behavior • Known effects of DVE on feedback sources Model of Current Weather Conditions • Ambient air temperature • Wind speed • Visibility • <Other conditions as needed>	Assumes that there is no real-time way to determine if an instrument is compromised by DVE conditions and therefore a priori knowledge is required <Rationale for why each environmental condition is included and what instrument is anticipated to be affected by that condition>
SR-3.	Validate the inputs or feedback received from other controllers before using that input or feedback to update process models [SLR-3]	FOS will cross-check the feedback received from other controllers with another data source to confirm that the feedback from the other controller and the other data source agree before using the feedback	Model of System Behavior • Data sources used to maintain each parameter in the process model	Assumes that it is possible to cross-check all feedback within the time limits available and with the data available <Any assumptions related to the risk of confusion when multiple data sources are available or the risks associated with ignoring the input if it cannot be successfully cross-checked>

Resp. ID	Responsibility	Decision-Making Strategy	Required Process Model Contents	Rationale and Assumptions
SR-4.	Confirm that all aspects of the aircraft state match the expected safe departure state before providing actuator movements to depart [SLR-4]	FOS compares current aircraft state to expected safe departure state	Current Aircraft Mission Readiness • State of cargo load • State of personnel load Current Aircraft System State • State of all aircraft subsystems Expected safe departure state • Cargo required • Personnel required • Expected state of aircraft subsystems	<Rationale for why the piloting controller comparing current and expected safe departure state is chosen instead of alternative options (e.g. preflight personnel doing all the checks and then just informing the pilot)> <Rationale for why each feedback item needs to be checked>
SR-5.	Determine if the state of the aircraft changes between the time that it was checked and departure [SLR-5]	Based on the last time the aircraft state was compared to the expected safe departure state, if the aircraft system state is updated after that time, perform the comparison again	Expected Safe Departure State: • Last time state was compared to current aircraft state Current Aircraft System State: • Time of last update	

Resp. ID	Responsibility	Decision-Making Strategy	Required Process Model Contents	Rationale and Assumptions
SR-6.	Confirm that data about the expected safe departure state of the aircraft has been fully specified and received [SLR-6, SLR-9]	The FOS uses a standardized template of information that defines the expected safe departure state for a specific mission. The FOS should check that everything in the template is specified for the mission. The FOS should also check that all controllers who need to receive the information have successfully received it	Expected safe departure state • Template of information that should be specified for a mission • Controllers that need to know/modify this information • Confirmation of receipt by controllers	This assumes that the information needed to completely specify a mission can be written in terms of a template. If the definition of "completely specified" depends on the mission, then a more flexible method of defining the criteria for evaluating if a mission has been "completely specified" will need to be defined
SR-7.	Ensure that all relevant personnel are aware of any changes to the expected safe departure state of the aircraft [SLR-7]	When anything in the expected safe departure state changes after the time of last update, the FOS should ensure that it receives confirmation of receipt for the update from every controller	Expected safe departure state • Time of last update • Controllers that need to know/modify this information • Confirmation of receipt by controllers	

Resp. ID	Responsibility	Decision-Making Strategy	Required Process Model Contents	Rationale and Assumptions
SR-8.	Process all feedback to make a deliberate decision if it is to be ignored/dropped [SLR-8]	Perform the decision making strategies in SR-1, SR-2 and SR-3 to determine if any of those considerations result in a decision to ignore the data	All process model parts as listed in SR-1, SR-2 and SR-3	Assumes that data might only be ignored/dropped if it is: 1. Out of date 2. In conflict with other data sources 3. Considered untrustworthy under current DVE conditions
SR-9.	Ensure that all data needed to determine the state of the aircraft, state of the airspace and the environmental conditions around the aircraft under all DVE conditions is available at all times [SLR-9, SLR-15]	For each piece of feedback information required to maintain the process models, ensure that updated data is being received at the expected update frequency	Model of System Behavior • Expected update frequency for each input • Time since last update for each input • State of all data sources	<Rationale for the update frequency expected>

Resp. ID	Responsibility	Decision-Making Strategy	Required Process Model Contents	Rationale and Assumptions
SR-10.	Ensure that flight crews are aware of the most updated state of any maintenance tasks on the aircraft [SLR-10]	Confirmation of receipt should be received for any changes to maintenance tasks on the aircraft occurring after the time that a mission has been briefed to flight crews	Aircraft Mission Readiness • Time of mission briefing Current Aircraft State • Maintenance tasks on aircraft • Time of update to maintenance tasks • Confirmation of receipt of update	
SR-11.	Ensure that only authorized actuator movements are executed [SLR-11]	The authenticity and integrity of the actuator movements should be verified before they are executed using <TBD> cryptographic methods	Model of System Behavior • Keys associated with controllers on the aircraft	Assumes that risks of unauthorized actuator movements only occur after the control column sensors (e.g. unauthorized physical access to the control column is not considered) Key-based authentication methods are well-established methods of authenticating messages

Resp. ID	Responsibility	Decision-Making Strategy	Required Process Model Contents	Rationale and Assumptions
SR-12.	Be able to operate in the expected DVE conditions [SLR-12]	Compare <TBD metrics> based on expected worst-case DVE conditions and the conditions that the equipment is designed to operate in to ensure that the systems being used have been designed to operate in those conditions	Model of System Behavior • Worst case conditions that the equipment is designed to be used in • Metrics to determine suitability of equipment	<Rationale for the metrics chosen>
SR-13.	Distinguish useful feedback from noise in feedback data [SLR-13]	Use typical noise and expected signal strength under all DVE conditions to help isolate signal from noise and cross check with other data sources to confirm	Model of System Behavior • Typical noise expected for each data source under DVE condition • Expected signal strength for each data source under DVE condition • Data sources used to maintain each part of the process model	<Assumptions about minimum signal strengths> <Assumptions about the conditions under which a weak signal might be received> <Assumptions/rationale about typical noise based on noise assessments>

Resp. ID	Responsibility	Decision-Making Strategy	Required Process Model Contents	Rationale and Assumptions
SR-14.	Process sensor data and use it to update its process model sufficiently quickly [SLR-14]	Use data from <selected sensor types> selected for SR-18, detect the environmental conditions or objects and other aircraft in the environment within <TBD> seconds	All process model contents in Current Weather Conditions and Current Airspace State	<Rationale for the time available to perform the detections based on sensors selected in SR-18, reliability of detection, difficulty in making required detections and the time needed to make those detections>
SR-15.	Determine if the state of the airspace changes between the time that it was checked and the commencement of a maneuver [SLR-16]	If the process model was last updated more than TBD seconds prior to commencing actuator movements, the process model should be updated again	Current Airspace State: • Time of last feedback measurement • Threshold at which airspace state process model is out of date	<Rationale for how that threshold is chosen>

Resp. ID	Responsibility	Decision-Making Strategy	Required Process Model Contents	Rationale and Assumptions
SR-16.	Account for the current and future movements of other aircraft in the vicinity when selecting actuator movements [SLR-17]	Anticipate current and future position of other aircraft and avoid own aircraft being in the same location	Model of Current Airspace State • IDs, Positions and Speeds of other aircraft and objects Anticipated Future Airspace State • Intent (expected future movement) of other aircraft and objects • Own Aircraft Intent Model of Current Aircraft Navigation State • Current air speed • Current altitude • Current position	<Rationale for why this method of accounting for future movements of aircraft> Assumes intent can always be communicated before actuator movements need to be provided either by own aircraft or the other aircraft

Resp. ID	Responsibility	Decision-Making Strategy	Required Process Model Contents	Rationale and Assumptions
SR-17.	Ensure that aircraft movements are selected such that sufficient reaction time is available if intent or movements of other aircraft are not what was expected, even if no violation of minimum separation is initially expected [SLR-18]	The decision-making strategy for SR-16 should be carried out such that the closest point of approach is no less than the closest point of approach distance to ensure that there is sufficient time to change course if the other aircraft behaves in unanticipated ways	All process model parts in SR-16 and the following: Model of System Behavior • Closest point of approach distance	<Rationale for closest point of approach distance based on anticipated speeds, reaction times etc.>
SR-18.	Detect all objects and other aircraft in the environment under all DVE conditions at all times [SLR-19]	Use data from <sensor choices> to measure the position and speed of objects and aircraft in the airspace	All process model contents in Current Weather Conditions and Current Airspace State	<Rationale for sensor choices based on anticipated objects and DVE conditions that might be encountered, reliability of detection, difficulty in making required detections, time needed to make those detections and noise assumptions made in SR-13> <Assumptions about minimum object size and DVE conditions>

Resp. ID	Responsibility	Decision-Making Strategy	Required Process Model Contents	Rationale and Assumptions
SR-19.	Detect slow or subtle changes in the state of the aircraft under all DVE conditions at all times [SLR-20]	Based on the navigation tolerance requirements used to select landing zones and flight paths, the FOS must be able to navigate within those tolerances and detect changes in position, altitude and airspeed that exceed those tolerances so that they can be counteracted	Current Aircraft Navigation State • Current position • Current altitude • Current airspeed Model of System Behavior: • Navigation tolerances to be maintained during flight	<Assumptions/rationale for where the navigation accuracy assumptions come from> <Rationale for the accuracy thresholds>

Resp. ID	Responsibility	Decision-Making Strategy	Required Process Model Contents	Rationale and Assumptions
SR-20.	Select actuator movements that minimize the risk of violation of minimum separation when information about the state of the airspace is in a degraded condition (e.g. delayed) [SLR-21]	Based on the FOS' own aircraft intent, if it is possible to wait to receive the delayed feedback, the FOS should do so If not, based on the last determinations of the current airspace state and anticipated future airspace state, the FOS should select actuator movements using established guidelines for how to minimize the likelihood of violating minimum separation	Anticipated Future Airspace State: • Own aircraft intent • Intent of other aircraft and objects Current Airspace State: • IDs, Positions and speeds of other aircraft and objects Model of System Behavior • Established guidelines for minimizing risk of violating minimum separation	<Rationale for why reliance on established guidelines is necessary> Assumes that there will always be an apparent movement available that will minimize the likelihood of violating minimum separation

Resp. ID	Responsibility	Decision-Making Strategy	Required Process Model Contents	Rationale and Assumptions
SR-21.	Respond quickly and with appropriate magnitude to disturbances to prevent unintended movement of the aircraft [SLR-22]	Based on <anticipated worst case DVE conditions such as wind gusts or turbulence effects>, unintended movement must be detected within <some time or distance threshold> and an appropriate actuator movement selected based on <required responses to maintain stable flight>	Model of Current Weather Conditions • Outside air temperature • Wind speed • Visibility • <Other conditions as needed> Model of Current Aircraft Navigation State • Current air speed • Current altitude • Current position Model of System Behavior • Actuator movements selection guidelines for handling the aircraft in DVE conditions	Assumes that the goal of preventing unintended movement is always appropriate (as opposed to accommodating some unintended movement to reduce airframe stress) <Rationale for worst case DVE conditions and response thresholds>

Resp. ID	Responsibility	Decision-Making Strategy	Required Process Model Contents	Rationale and Assumptions
SR-22.	Respond quickly enough and with appropriate magnitude to select and effect the desired change in flight path under the given environmental conditions [SLR-23, SLR-25]	Normal decision making process needed to select actuator movements, however decision making must happen more quickly	Model of Current Weather Conditions • Outside air temperature • Wind speed • Visibility • <Other conditions as needed> Model of Current Aircraft Navigation State • Current air speed • Current altitude • Current position Model of System Behavior • Actuator movements selection guidelines for handling the aircraft in DVE conditions	Assumes the "rules" for selecting actuator movements will be the same whether in DVE conditions or not but those decisions will need to be more responsive to disturbances caused by some DVE conditions than normal
SR-23.	Ensure that a viable flight path is always available to avoid any violations of minimum separation [SLR-24]	Based on current and anticipated airspace state, select a flight path that offers the most options for deviation/change if other violations of minimum separation occur	Model of Current Airspace State • IDs, Positions and speeds of other aircraft and objects Anticipated Future Airspace State • Intent (expected future movement) of other aircraft and objects	

Resp. ID	Responsibility	Decision-Making Strategy	Required Process Model Contents	Rationale and Assumptions
SR-24.	Select a viable flight path that avoids all possible violations of minimum separation [SLR-26]	Based on current and anticipated airspace state, select a flight path that accounts for all currently known possible violations of minimum separation	Model of Current Airspace State • IDs, Positions and Speeds of other aircraft and objects Anticipated Future Airspace State • Intent (expected future movement) of other aircraft and objects	

Sources of Feedback

Table 50: Full Details for Feedback Sources

Process Model Sub-Part	Required Information	Source of Information	Rationale/assumptions for source of information
Current Aircraft Mission Readiness	State of cargo load	Piloting Controller or preflight personnel	Some cargo might be loaded by maintenance or preflight personnel but others might be loaded by crew. Therefore, it is simplest for pilot to ensure that everything is on board regardless of who loaded it
	State of Personnel load	Piloting Controller	To determine the presence of personnel on board, it is easiest for the piloting controller to determine that on its own
	Time of mission briefing	EMS Operations and ATC	EMS Operations and ATC are the ones who are plan and brief the mission so they would know the time that the briefing was given

Process Model Sub-Part	Required Information	Source of Information	Rationale/assumptions for source of information
	Time of last feedback measurement	Piloting Controller	Since either the Piloting Controller checks the state of the aircraft, they know what time they checked it
Current Aircraft System State	State of all aircraft subsystems	Aircraft Subsystems	These values are measured by sensors on the aircraft
	Time of last feedback measurement	Aircraft Subsystems	
	Maintenance Tasks on the Aircraft	Maintenance Personnel	Since maintenance personnel are in charge of maintenance tasks, they know best the maintenance state of each aircraft and when they last updated maintenance tasks
	Time of update to maintenance tasks		
	Confirmation of receipt of update	Piloting Controller	Since the Piloting Controller received the update, they know what time they received it
Current Aircraft Navigation State	Current Air Speed	Aircraft subsystems	These values are measured by sensors on the aircraft
	Current Altitude		
	Current Position		
Expected Safe Departure State	Expected state of aircraft subsystems	Maintenance Personnel	Since maintenance personnel are in charge of maintenance tasks, they know best the maintenance state of the aircraft
	Time of Last Comparison to Current Aircraft State	Piloting Controller	Since either the Piloting Controller checks the state of the aircraft, they know what time they checked it
	Time of last update	EMS Operations and ATC Maintenance Personnel	Since EMS Operations and ATC plans the mission, they know what is required for the mission
	Cargo required	EMS Operations and ATC	
	Personnel required		
	Template of mission information		

Process Model Sub-Part	Required Information	Source of Information	Rationale/assumptions for source of information
	Controllers that need to know/modify safe departure state information		Since EMS Operations and ATC plans the mission, they know what is required for the mission
	Confirmation of receipt of departure state information		
Current Weather Conditions	Ambient Air Temperature	Aircraft Subsystems	These values are measured by sensors on the aircraft
	Wind Speed		
	Visibility		
Current Airspace State	IDs, Positions and Speeds of other aircraft and objects	Aircraft Subsystems Reports from other controllers EMS Operations and ATC	These values are expected to be determined using on-board sensor data but may also be determined by feedback from other controllers such as EMS Operations and ATC or other aircraft
	Time of last feedback measurement	Aircraft Subsystems	
	Threshold at which airspace state process model is out-of-date	EMS Operations and ATC	Since EMS Operations and ATC plan missions and define best practices, they know when airspace state information should be considered out of date
Anticipated Future Airspace State	Intent of other aircraft	Other aircraft	
	Own aircraft intent	Piloting Controller	
Model of System Behavior	Expected update frequency for each input		Since EMS Operations and ATC knows aircraft capabilities, they can provide this feedback

Process Model Sub-Part	Required Information	Source of Information	Rationale/assumptions for source of information
	Time since last update for each input	EMS Operations and ATC/Maintenance Personnel	Since all of this information is related to the mission being planned and the equipment needed, the feedback comes from EMS Operations and ATC
	Threshold for out-of-date data	EMS Operations and ATC	
	Known effects of DVE on feedback sources		
	Data sources used to maintain each part of the process model		
	Keys associated with controllers on the aircraft		
	Worst case conditions that the equipment is designed to be used in		
	Metrics to determine suitability of equipment		
	Typical noise expected for each data source under DVE conditions		
	Expected signal strength for each data source under DVE conditions		
	Navigation tolerances to be maintained during flight		
	Closest point of approach distance		

Process Model Sub-Part	Required Information	Source of Information	Rationale/assumptions for source of information
	Established guidelines for minimizing risk of violating minimum separation		<Rationale for the guidelines>
	Actuator movements selection guidelines for handling the aircraft in DVE conditions		Since all of this information is related to the mission being planned and the equipment needed, the feedback comes from EMS Operations and ATC
	State of all data sources		

Timing Requirements

To show how timing requirements should be defined and documented, Table 51 shows the timing requirements for the non-feedback validation responsibilities. Although not shown, timing requirements should also be defined for the feedback validation responsibilities as well.

Non-Feedback-Validation Responsibilities

Table 51: Full Details for Timing Requirements for Non-Feedback-Validation Responsibilities

Resp. ID	Responsibility	Timing Requirements	Rationale/Assumptions
SR-4	Confirm that all aspects of the aircraft state match the expected safe departure state before providing actuator movements to depart [SLR-4]	Occurs once per mission <The shortest amount of time that would be available to perform this responsibility in an emergency deployment>	<Rationale or assumptions about how the emergency deployment scenario was defined that led to the timing requirements>
SR-5	Determine if the state of the aircraft changes between the time that it was checked and departure [SLR-5]		

Resp. ID	Responsibility	Timing Requirements	Rationale/Assumptions
SR-6	Confirm that data about the expected safe departure state of the aircraft has been fully specified and received [SLR-6, SLR-9]		
SR-7	Ensure that all relevant personnel are aware of any changes to the expected safe departure state of the aircraft [SLR-7]	Occurs <TBD> per deployment based on historical mission data <Based on shortest time available in an emergency deployment, determine how quickly such changes need to be disseminated	<Rationale for how historical mission data was used to determine frequency of occurrence> <Rationale for how the speed of dissemination of information must occur>
SR-10	Ensure that flight crews are aware of the most updated state of any maintenance tasks on the aircraft [SLR-10]		
SR-11	Ensure that only authorized actuator movements are executed [SLR-11]	Many times per mission <Determine limitation based on latency requirements of aircraft control system>	<Rationale for or link to latency requirements of aircraft control system>
SR-12	Be able to operate in the expected DVE conditions [SLR-12]	Occurs once per mission <The shortest amount of time that would be available to perform this responsibility in an emergency deployment>	<Rationale or assumptions about how the emergency deployment scenario was defined that led to the timing requirements>

Resp. ID	Responsibility	Timing Requirements	Rationale/Assumptions
SR-15	Determine if the state of the airspace changes between the time that it was checked and the commencement of a maneuver [SLR-16]	Many times per mission <Determine limitation based on latency requirements of aircraft control system>	<Rationale for or link to latency requirements of aircraft control system>
SR-16	Account for the current and future movements of other aircraft in the vicinity when selecting actuator movements	<Timing requirements here would be defined based on the most urgent scenario expected to be encountered and how quickly the piloting controller would need to select actuator movements in that scenario>	<Rationale about the scenario used to define this timing constraint>
SR-17	Ensure that aircraft movements are selected such that sufficient reaction time is available if intent or movements of other aircraft are not what was expected, even if no violation of minimum separation is initially expected [SLR-18]		
SR-18	Detect all objects and other aircraft in the environment under all DVE conditions at all times [SLR-19]		
SLR-19	Detect slow or subtle changes in the state of the aircraft under all DVE conditions at all times [SLR-21]		
SR-20	Select actuator movements that minimize the risk of violation of minimum separation when information about the state of the airspace is in a degraded condition (e.g. delayed) [SLR-21]		

Resp. ID	Responsibility	Timing Requirements	Rationale/Assumptions
SR-21	Respond quickly and with appropriate magnitude to disturbances to prevent unintended movement of the aircraft	<Timing requirements here would be defined by aircraft dynamics and effects of DVE conditions>	<Rationale about aircraft dynamics and DVE conditions being considered>
SR-22	Respond quickly enough and with appropriate magnitude to select and effect the desired change in flight path under the given environmental conditions	<Similar to R-3.1.3, aircraft dynamics and DVE conditions would define timing requirements>	<Rationale about aircraft dynamics and DVE conditions being considered>
SR-23	Ensure that a viable flight path is always available to avoid any violations of minimum separation	<Timing constraint that defines how often the piloting controller should re-evaluate whether they have a viable flight path options should they need to avoid a collision> <The shortest amount of time that would be available to perform this flight path evaluation>	<Rationale/assumptions that led to defining how often the piloting controller should re-evaluate whether they have viable flight path options>
SR-24	Select a viable flight path that avoids all possible violations of minimum separation	<Timing constraint that defines how often the piloting controller should re-evaluate that their flight path is still valid> <The shortest amount of time that would be available to perform this flight path evaluation>	<Rationale/assumptions that led to defining how often the piloting controller should re-evaluate their flight path>

Appendix E Responsibility Assignments for Part 1 Architecture Option 1

This appendix shows the full set of responsibility assignments for part 1 architecture option 1 for both feedback validation responsibilities and non-feedback-validation responsibilities.

Non-Feedback-Validation Responsibilities

Table 52: Non-Feedback-Validation Responsibility Assignments for Part 1 Architecture Option 1

Resp. ID	Responsibility	Assigned to				Rationale/Assumptions
		EMSA TC	MP	PC	AS	
SR-4	Confirm that all aspects of the aircraft state match the expected safe departure state before providing actuator movements to depart [SLR-4]			X		Since the piloting controller is primarily responsible for controlling the aircraft and executing the mission, this related responsibility for making final confirmation that the aircraft is in a safe departure state before departing should also be assigned to the piloting controller
SR-5	Determine if the state of the aircraft changes between the time that it was checked and departure [SLR-5]			X		Since this is related to SR-4, for the same reasons that SR-4 is assigned to the piloting controller, so should this one
SR-6	Confirm that data about the expected safe departure state of the aircraft has been fully specified and received [SLR-6, SLR-9]	X		X		Since the expected safe departure state is dependent on the needs of a specific mission, EMS Operations and ATC has to be assigned responsibility for planning the mission. However, the piloting controller could also confirm that the information it receives is complete

Resp. ID	Responsibility	Assigned to				Rationale/Assumptions
		EMSA TC	MP	PC	AS	
SR-7	Ensure that all relevant personnel are aware of any changes to the expected safe departure state of the aircraft [SLR-7]	X				Assuming that changes are likely to either come from EMS Operations and ATC or need to be approved by EMS Operations and ATC, then EMS Operations and ATC will be informed of any changes that are needed and therefore is in the best position to disseminate any changes
SR-10	Ensure that flight crews are aware of the most updated state of any maintenance tasks on the aircraft [SLR-10]		X			Similar to SR-7, assuming that changes to the state of maintenance tasks either come from the maintenance personnel or must be raised with the maintenance personnel, the maintenance personnel are therefore in the best position to disseminate any changes to maintenance tasks to flight crews
SR-11	Ensure that only authorized actuator movements are executed [SLR-11]				X	Since the execution of actuator movements to move actuators is performed by the aircraft subsystems, the responsibility for only executing authorized actuator movements is assigned to the aircraft subsystems
SR-12	Be able to operate in the expected DVE conditions [SLR-12]	X				Since EMS Operations and ATC is responsible for planning a mission, that planning should include selection of appropriate equipment, thus this responsibility is assigned to EMS Operations and ATC

Resp. ID	Responsibility	Assigned to				Rationale/Assumptions
		EMSA TC	MP	PC	AS	
SR-15	Determine if the state of the airspace changes between the time that it was checked and the commencement of a maneuver [SLR-16]			X		Since the piloting controller is responsible for controlling the aircraft and this responsibility relates to how to maintain safe control of the aircraft, it is assigned to the piloting controller
SR-16	Account for the current and future movements of other aircraft in the vicinity when selecting actuator movements [SLR-17]			X		All of these responsibilities relate to selecting appropriate flight paths and actuator movements, all of which are within the scope of the piloting controller's responsibility for maintaining safe control over the aircraft. As such, these responsibilities are all assigned to the piloting controller
SR-17	Ensure that aircraft movements are selected such that sufficient reaction time is available if intent or movements of other aircraft are not what was expected, even if no violation of minimum separation is initially expected [SLR-18]			X		
SR-18	Detect all objects and other aircraft in the environment under all DVE conditions at all times [SLR-19]			X	X	Under DVE conditions, the piloting controller's direct observations will need to be augmented by a sensor suite that is part of the aircraft subsystems
SR-19	Detect slow or subtle changes in the state of the aircraft under all DVE conditions at all times [SLR-20]			X		All of these responsibilities relate to selecting appropriate flight paths and actuator movements, all of which are within the scope of the piloting controller's responsibility for

Resp. ID	Responsibility	Assigned to				Rationale/Assumptions
		EMSA TC	MP	PC	AS	
SR-20	Select actuator movements that minimize the risk of violation of minimum separation when information about the state of the airspace is in a degraded condition (e.g. delayed) [SLR-21]			X		maintaining safe control over the aircraft. As such, these responsibilities are all assigned to the piloting controller
SR-21	Respond quickly and with appropriate magnitude to disturbances to prevent unintended movement of the aircraft [SLR-22]			X		
SR-22	Respond quickly enough and with appropriate magnitude to select and effect the desired change in flight path under the given environmental conditions [SLR-23, SLR-25]			X		
SR-23	Ensure that a viable flight path is always available to avoid any violations of minimum separation [SLR-24]			X		
SR-24	Select a viable flight path that avoids all possible violations of minimum separation [SLR-26]			X		

Feedback Validation Responsibilities

Table 53: Feedback Validation Responsibilities for Part 1 Architecture Option 1

<table>
<tr><th>Resp ID</th><th>Responsibility</th><th>Assignments for Each Process Model Part</th><th>Rationale/Assumptions</th></tr>
<tr><td>SR-1</td><td>Determine if feedback received about the aircraft's mission readiness, state of the aircraft and the airspace is too old [SLR-1]</td><td><table>
<tr><td>Aircraft Mission Readiness</td><td>PC</td></tr>
<tr><td>Current Aircraft System State</td><td>PC</td></tr>
<tr><td>Current Aircraft Navigation State</td><td>PC</td></tr>
<tr><td>Expected Safe Departure State</td><td>PC</td></tr>
<tr><td>Current Weather Conditions</td><td>PC</td></tr>
<tr><td>Current Airspace State</td><td>PC</td></tr>
<tr><td>Anticipated Future Airspace State</td><td>PC</td></tr>
<tr><td>Model of System Behavior</td><td>EMSATC</td></tr>
</table></td><td>Since most of these process models are used to fly the aircraft or select a flight path, the piloting controller is in the best position to decide when feedback is too old.

For the model of system behavior, EMSATC is assigned because it is that they will evaluate feedback they might receive to determine when those models may need to be updated</td></tr>
<tr><td>SR-2</td><td>Account for prevailing DVE conditions and their possible effect on feedback sources when making use of feedback information [SLR-2]</td><td><table>
<tr><td>Aircraft Mission Readiness</td><td>N/A</td></tr>
<tr><td>Current Aircraft System State</td><td>PC</td></tr>
<tr><td>Current Aircraft Navigation State</td><td>PC</td></tr>
<tr><td>Expected Safe Departure State</td><td>N/A</td></tr>
<tr><td>Current Weather Conditions</td><td>PC</td></tr>
<tr><td>Current Airspace State</td><td>PC</td></tr>
<tr><td>Anticipated Future Airspace State</td><td>PC</td></tr>
<tr><td>Model of System Behavior</td><td>N/A</td></tr>
</table></td><td>Since this responsibility is dependent on prevailing weather conditions during the flight, this responsibility must be carried repeatedly and in real-time as discussed in the timing requirements section. As such, the piloting controller is the only controller equipped to do this.

This assumes that feedback for aircraft mission readiness, expected safe departure state and model of system behavior will not be affected by DVE and hence this responsibility does not apply for those process model parts.</td></tr>
</table>

<table>
<tr><th>Resp ID</th><th>Responsibility</th><th>Assignments for Each Process Model Part</th><th>Rationale/Assumptions</th></tr>
<tr><td>SR-3</td><td>Validate the inputs or feedback received from other controllers before using that input or feedback to update process models [SLR-3]</td><td><table>
<tr><td>Aircraft Mission Readiness</td><td>PC</td></tr>
<tr><td>Current Aircraft System State</td><td>PC</td></tr>
<tr><td>Current Aircraft Navigation State</td><td>PC</td></tr>
<tr><td>Expected Safe Departure State</td><td>EMSATC</td></tr>
<tr><td>Current Weather Conditions</td><td>PC</td></tr>
<tr><td>Current Airspace State</td><td>PC</td></tr>
<tr><td>Anticipated Future Airspace State</td><td>PC</td></tr>
<tr><td>Model of System Behavior</td><td>EMSATC</td></tr>
</table></td><td>Similar to SR-1, the piloting controller is in the best position to decide when feedback is too old since these process model parts are used to fly the aircraft or select a flight path.

This includes the model of system behavior which could have its parameters updated based on input that needs to be validated</td></tr>
<tr><td>SR-8</td><td>Process all feedback to make a deliberate decision if it is to be ignored/dropped [SLR-8]</td><td><table>
<tr><td>Aircraft Mission Readiness</td><td>PC</td></tr>
<tr><td>Current Aircraft System State</td><td>PC</td></tr>
<tr><td>Current Aircraft Navigation State</td><td>PC</td></tr>
<tr><td>Expected Safe Departure State</td><td>EMSATC</td></tr>
<tr><td>Current Weather Conditions</td><td>PC</td></tr>
<tr><td>Current Airspace State</td><td>PC</td></tr>
<tr><td>Anticipated Future Airspace State</td><td>PC</td></tr>
<tr><td>Model of System Behavior</td><td>EMSATC</td></tr>
</table></td><td>Similar to SR-1, the piloting controller is in the best position to decide when feedback is too old since these process model parts are used to fly the aircraft or select a flight path.

This is marked as N/A for the model of system behavior because it is assumed that the piloting controller will not get real-time feedback about the model of system behavior that will need to be validated</td></tr>
</table>

<table>
<tr><th>Resp ID</th><th>Responsibility</th><th>Assignments for Each Process Model Part</th><th>Rationale/Assumptions</th></tr>
<tr><td>SR-9</td><td>Ensure that all data needed to determine the state of the aircraft, state of the airspace and the environmental conditions around the aircraft under all DVE conditions is available at all times [SLR-9, SLR-15]</td><td><table>
<tr><td>Aircraft Mission Readiness</td><td>PC</td></tr>
<tr><td>Current Aircraft System State</td><td>PC</td></tr>
<tr><td>Current Aircraft Navigation State</td><td>PC</td></tr>
<tr><td>Expected Safe Departure State</td><td>N/A</td></tr>
<tr><td>Current Weather Conditions</td><td>PC</td></tr>
<tr><td>Current Airspace State</td><td>PC</td></tr>
<tr><td>Anticipated Future Airspace State</td><td>PC</td></tr>
<tr><td>Model of System Behavior</td><td>N/A</td></tr>
</table></td><td rowspan="2">Similar to SR-2, these responsibilities have to be performed repeatedly during flight as described in the timing requirements section. As such, the piloting controller is the only controller in the system that is equipped to do this.

These assignments assume that continuous feedback is not expected for the expected safe departure state and model of system and hence this responsibility does not apply for those process model parts.</td></tr>
<tr><td>SR-13</td><td>Distinguish useful feedback from noise in feedback data [SLR-13]</td><td><table>
<tr><td>Aircraft Mission Readiness</td><td>PC</td></tr>
<tr><td>Current Aircraft System State</td><td>PC</td></tr>
<tr><td>Current Aircraft Navigation State</td><td>PC</td></tr>
<tr><td>Expected Safe Departure State</td><td>N/A</td></tr>
<tr><td>Current Weather Conditions</td><td>PC</td></tr>
<tr><td>Current Airspace State</td><td>PC</td></tr>
<tr><td>Anticipated Future Airspace State</td><td>PC</td></tr>
<tr><td>Model of System Behavior</td><td>N/A</td></tr>
</table></td></tr>
<tr><td>SR-14</td><td>Process sensor data and use it to update its process model sufficiently quickly [SLR-14]</td><td><table>
<tr><td>Aircraft Mission Readiness</td><td>PC</td></tr>
<tr><td>Current Aircraft System State</td><td>PC</td></tr>
<tr><td>Current Aircraft Navigation State</td><td>PC</td></tr>
<tr><td>Expected Safe Departure State</td><td>EMSATC</td></tr>
<tr><td>Current Weather Conditions</td><td>PC</td></tr>
<tr><td>Current Airspace State</td><td>PC</td></tr>
<tr><td>Anticipated Future Airspace State</td><td>PC</td></tr>
<tr><td>Model of System Behavior</td><td>EMSATC</td></tr>
</table></td><td>Similar to SR-2, these responsibilities have to be performed repeatedly during flight as described in the timing requirements section. As such, the piloting controller is the only controller in the system that is equipped to do this.

These assignments assume that continuous feedback is not expected for the expected safe departure state and model of system and hence this responsibility does not apply for those process model parts.</td></tr>
</table>

Appendix F Responsibility Assignments for Part 2 Architecture Option 1

Assignment of Non-Feedback-Validation Responsibilities

Table 54: Assignment of Non-Feedback-Validation Responsibilities for Part 2 Architecture Option 1

Resp. ID	Responsibility	Assigned to		Rationale/Assumptions
		Pilot	ASEC	
SR-4	Confirm that all aspects of the aircraft state match the expected safe departure state before providing actuator movements to depart [SLR-4]	X		Expected safe departure state is most easily obtained by human pilot instead of ASEC
SR-5	Determine if the state of the aircraft changes between the time that it was checked and departure [SLR-5]	X		Since the state of the aircraft requires consideration of more than just sensor inputs, the human pilot is better positioned to interpret these other inputs than automation
SR-6	Confirm that data about the expected safe departure state of the aircraft has been fully specified and received [SLR-6, SLR-9]	X		Since the human pilot will be checking the aircraft against the safe departure state, it is convenient for them to also check that the safe departure state is fully specified so that they can clarify any uncertainties if necessary
SR-15	Determine if the state of the airspace changes between the time that it was checked and the commencement of a maneuver [SLR-16]	X	X	Since this is relatively simple checking of the maneuver path, the ASEC is able to warn the human pilot of potential collisions but the human pilot is expected to perform an independent verification as well
SR-16	Account for the current and future movements of other aircraft in the vicinity when selecting actuator movements [SLR-17]	X		Coordinating with other aircraft and adapting to work with their movements is more easily done in real-time by a human than automation
SR-17	Ensure that aircraft movements are selected such that sufficient reaction time is available if intent or movements of other aircraft are not what was expected, even if no violation of minimum separation is initially expected [SLR-18]	X		Since this involves integrating intent information alongside current airspace state and aircraft state information to decide the best path to take and the best actuator movements to apply, this is best done by a human pilot

Resp. ID	Responsibility	Assigned to		Rationale/Assumptions
		Pilot	ASEC	
SR-18	Detect all objects and other aircraft in the environment under all DVE conditions at all times [SLR-19]	X	X	This is shared because the ASEC can provide the human pilot with detection information but the human pilot can also perform independent visual identification (when conditions allow) or integrate other types of data sources to verify a detection. Hence, this responsibility is shared between the human pilot and ASEC
SR-19	Detect slow or subtle changes in the state of the aircraft under all DVE conditions at all times [SLR-20]		X	Given the potential magnitude of these changes in the state of the aircraft, it may be difficult to design appropriate feedback mechanisms with sufficient saliency for a human pilot to notice those changes and they would be easier detected by automation
SR-20	Select actuator movements that minimize the risk of violation of minimum separation when information about the state of the airspace is in a degraded condition (e.g. delayed) [SLR-21]	X		Similar to SR-17, this decision making is complex enough and the definition of risk minimization is likely vague enough that it is better performed by a human pilot than by automation
SR-21	Respond quickly and with appropriate magnitude to disturbances to prevent unintended movement of the aircraft [SLR-22]	X	X	Assumes that the ASEC will not be able to stabilize the aircraft sufficiently under all DVE conditions and that a human pilot would respond more appropriately in some situations Also assumes that ASEC will be able to help stabilize the aircraft under limited DVE conditions and therefore the ASEC will sometimes be used
SR-22	Respond quickly enough and with appropriate magnitude to select and effect the desired change in flight path under the given environmental conditions [SLR-23, SLR-25]	X	X	
SR-23	Ensure that a viable flight path is always available to avoid any violations of minimum separation [SLR-24]	X		Assumes that automation sophistication will not be sufficient to select flight paths under all conditions and therefore it makes more sense for a human pilot to retain this function
SR-24	Select a viable flight path that avoids all possible violations of minimum separation [SLR-26]	X		Assumes that automation sophistication will not be sufficient to select flight paths under all conditions and therefore it makes more sense for a human pilot to retain this function

Assignment of Feedback Validation Responsibilities

Table 55: Assignment of Feedback Validation Responsibilities for Part 2 Architecture Option 1

<table>
<tr><th>Resp ID</th><th>Responsibility</th><th>Assignments for Each Process Model Part</th><th>Rationale/Assumptions</th></tr>
<tr><td>SR-1</td><td>Determine if feedback received about the aircraft's mission readiness, state of the aircraft and the airspace is too old [SLR-1]</td><td><table>
<tr><td>Aircraft Mission Readiness</td><td>Pilot</td></tr>
<tr><td>Current Aircraft System State</td><td>ASEC</td></tr>
<tr><td>Current Aircraft Navigation State</td><td>ASEC</td></tr>
<tr><td>Expected Safe Departure State</td><td>Pilot</td></tr>
<tr><td>Current Weather Conditions</td><td>ASEC</td></tr>
<tr><td>Current Airspace State</td><td>ASEC</td></tr>
<tr><td>Anticipated Future Airspace State</td><td>Pilot</td></tr>
<tr><td>Model of System Behavior</td><td>N/A</td></tr>
</table></td><td>Assuming the threshold is fixed or is easily determined based on simple interpretations of environmental factors or state of aircraft, this responsibility is repetitive and would be easily accomplished by automation

Mission readiness, safe departure states and anticipated future airspace state are more easily received by the pilot, hence the pilot should check these rather than the aircraft</td></tr>
<tr><td>SR-2</td><td>Account for prevailing DVE conditions and their possible effect on feedback sources when making use of feedback information [SLR-2]</td><td><table>
<tr><td>Aircraft Mission Readiness</td><td>N/A</td></tr>
<tr><td>Current Aircraft System State</td><td>ASEC</td></tr>
<tr><td>Current Aircraft Navigation State</td><td>Pilot</td></tr>
<tr><td>Expected Safe Departure State</td><td>N/A</td></tr>
<tr><td>Current Weather Conditions</td><td>Pilot</td></tr>
<tr><td>Current Airspace State</td><td>Pilot</td></tr>
<tr><td>Anticipated Future Airspace State</td><td>Pilot</td></tr>
<tr><td>Model of System Behavior</td><td>N/A</td></tr>
</table></td><td>Mission readiness and expected safe departure state is communicated verbally to the pilot and therefore is unaffected by DVE. Similarly, current aircraft system state is not affected by DVE since it is a detection of internal aircraft system states.

Assumes that the determinations involved are more quickly made by a human than for those rules to be implemented in automation</td></tr>
</table>

<table>
<tr><th>Resp ID</th><th>Responsibility</th><th>Assignments for Each Process Model Part</th><th>Rationale/Assumptions</th></tr>
<tr><td>SR-3</td><td>Validate the inputs or feedback received from other controllers before using that input or feedback to update process models [SLR-3]</td><td><table>
<tr><td>Aircraft Mission Readiness</td><td>Pilot</td></tr>
<tr><td>Current Aircraft System State</td><td>ASEC</td></tr>
<tr><td>Current Aircraft Navigation State</td><td>Pilot</td></tr>
<tr><td>Expected Safe Departure State</td><td>N/A</td></tr>
<tr><td>Current Weather Conditions</td><td>Pilot</td></tr>
<tr><td>Current Airspace State</td><td>Pilot</td></tr>
<tr><td>Anticipated Future Airspace State</td><td>Pilot</td></tr>
<tr><td>Model of System Behavior</td><td>N/A</td></tr>
</table></td><td>The variety of possible inputs made by other aircraft or controllers would be more easily handled by a human, except for the aircraft system state which would be more easily handled by automation since the system state is already generated and monitored by automation

Expected safe departure state and model of system behavior are marked as N/A because they were not assigned to the Piloting Controller in part 1</td></tr>
<tr><td>SR-8</td><td>Process all feedback to make a deliberate decision if it is to be ignored/dropped [SLR-8]</td><td><table>
<tr><td>Aircraft Mission Readiness</td><td>Pilot</td></tr>
<tr><td>Current Aircraft System State</td><td>ASEC</td></tr>
<tr><td>Current Aircraft Navigation State</td><td>Pilot</td></tr>
<tr><td>Expected Safe Departure State</td><td>Pilot</td></tr>
<tr><td>Current Weather Conditions</td><td>Pilot</td></tr>
<tr><td>Current Airspace State</td><td>Pilot</td></tr>
<tr><td>Anticipated Future Airspace State</td><td>Pilot</td></tr>
<tr><td>Model of System Behavior</td><td>Pilot</td></tr>
</table></td><td>As stated in the system-level behavior, this responsibility involves carrying out the decision making strategies of SR-1, SR-2 and SR-3. Since most of these are carried out by the Pilot, the pilot has responsibility for deciding to keep or drop a piece of feedback. The only exception is the aircraft system state where the ASEC is assigned this responsibility for that process model</td></tr>
</table>

<table>
<tr><th>Resp ID</th><th>Responsibility</th><th>Assignments for Each Process Model Part</th><th>Rationale/Assumptions</th></tr>
<tr><td>SR-9</td><td>Ensure that all data needed to determine the state of the aircraft, state of the airspace and the environmental conditions around the aircraft under all DVE conditions is available at all times [SLR-9, SLR-15]</td><td><table>
<tr><td>Aircraft Mission Readiness</td><td>Pilot</td></tr>
<tr><td>Current Aircraft System State</td><td>ASEC</td></tr>
<tr><td>Current Aircraft Navigation State</td><td>ASEC</td></tr>
<tr><td>Expected Safe Departure State</td><td>N/A</td></tr>
<tr><td>Current Weather Conditions</td><td>ASEC</td></tr>
<tr><td>Current Airspace State</td><td>ASEC</td></tr>
<tr><td>Anticipated Future Airspace State</td><td>Pilot</td></tr>
<tr><td>Model of System Behavior</td><td>N/A</td></tr>
</table></td><td>Monitoring the inputs for aircraft system state, navigation state and environmental conditions is a repetitive task best done by automation.

For anticipated future airspace state and aircraft mission readiness, since that feedback may include verbal radio communication, the pilot shares the responsibility to ensure that needed information is available.

Expected safe departure state and model of system behavior are not updated based on real-time data and are therefore N/A</td></tr>
<tr><td>SR-13</td><td>Distinguish useful feedback from noise in feedback data [SLR-13]</td><td><table>
<tr><td>Aircraft Mission Readiness</td><td>N/A</td></tr>
<tr><td>Current Aircraft System State</td><td>ASEC</td></tr>
<tr><td>Current Aircraft Navigation State</td><td>ASEC</td></tr>
<tr><td>Expected Safe Departure State</td><td>N/A</td></tr>
<tr><td>Current Weather Conditions</td><td>ASEC</td></tr>
<tr><td>Current Airspace State</td><td>Pilot
ASEC</td></tr>
<tr><td>Anticipated Future Airspace State</td><td>Pilot</td></tr>
<tr><td>Model of System Behavior</td><td>N/A</td></tr>
</table></td><td>Especially when identifying weak signals, automation is better at the pattern recognition necessary to extract weak signals

Assumes that the risk of false negative detections of weak signals is acceptably low

Mission readiness, expected safe departure state and model of system behavior do not rely on real-time feedback and therefore are not affected by noise</td></tr>
</table>

<table>
<tr><th>Resp ID</th><th>Responsibility</th><th>Assignments for Each Process Model Part</th><th>Rationale/Assumptions</th></tr>
<tr><td>SR-14</td><td>Process sensor data and use it to update its process model sufficiently quickly [SLR-14]</td><td>
<table>
<tr><td>Aircraft Mission Readiness</td><td>Pilot</td></tr>
<tr><td>Current Aircraft System State</td><td>ASEC</td></tr>
<tr><td>Current Aircraft Navigation State</td><td>Pilot
ASEC</td></tr>
<tr><td>Expected Safe Departure State</td><td>N/A</td></tr>
<tr><td>Current Weather Conditions</td><td>Pilot
ASEC</td></tr>
<tr><td>Current Airspace State</td><td>Pilot
ASEC</td></tr>
<tr><td>Anticipated Future Airspace State</td><td>Pilot</td></tr>
<tr><td>Model of System Behavior</td><td>N/A</td></tr>
</table>
</td><td>For everything but the aircraft system state, the pilot is primarily the one updating the process model parts and the ASEC only assists in the aircraft navigation state, weather conditions and airspace state.

Expected safe departure state is not updated using sensors and therefore is N/A

Expected safe departure state and model of system behavior are marked as N/A because they were not assigned to the Piloting Controller in part 1</td></tr>
</table>

Appendix G Additional STPA Analysis for Architecture Option 3

In this appendix, additional examples of UCAs and scenarios from the STPA analysis conducted for the new control actions introduced as part of Architecture Option 3 in Part 2 of the architecture creation process are presented.

Revised STPA Analysis of ASEC Actuator Movements Control Action

UCA-1.2: ASEC provides actuator movements that steers the aircraft toward another aircraft or object

CS-G-1.2.1: The ASEC provides actuator movements that steers the aircraft toward another aircraft or object despite receiving feedback that there was another aircraft or object nearby. This could occur because:

1. The ASEC may have the wrong process model of the state of the environment, the aircraft or the airspace around the aircraft and therefore wrongly believes that the nearby airspace is clear to pilot toward and plans a trajectory or path toward the other object or aircraft. Although the ASEC coordinates with the human pilot to confirm the trajectory that it planned, the trajectory planned by the ASEC is confirmed without modification. As a result, the erroneous trajectory planned by the ASEC is not corrected and the ASEC proceeds to execute the planned trajectory, believing the human pilot agreed with its plan. The ASEC might receive confirmation of its erroneously planned trajectory because:
   1.1. The human pilot and ASEC might have the same shared process model of the current state of the airspace. As a result, like the ASEC, the human pilot also does not recognize that the planned trajectory is headed toward a nearby object or aircraft and therefore does not correct the trajectory and agrees with the ASEC. The reasons that this might occur are described in scenarios CS-1.1.2-1 and CS-1.1.2-2 that were generated for UCAs issued by the human pilot.
   1.2. Alternatively, the human pilot and ASEC might not have the same shared process model of the current state of the airspace. If the human pilot has a different and correct process model of the state of the airspace, the human pilot might recognize the error in the planned trajectory and correctly modify it but the correctly modified trajectory is not received by the ASEC and instead the ASEC is told the human pilot agreed with its erroneous trajectory. The reasons that this might occur are described in scenarios CS-1.1.3-1 and CS-1.1.3-2 that were generated for UCAs issued by the human pilot.
2. The human pilot might also have a different but also erroneous or incomplete process model of the state of the airspace. Under these conditions, the human pilot might still correctly recognize the error in the ASEC's planned trajectory. However, although the modifications that the human pilot makes addresses the error that the ASEC made, the human pilot's modified trajectory now pilots the aircraft toward a different aircraft or object in the airspace instead. The reasons that this might occur are described in scenarios CS-1.8.1-1, CS-1.8.1-2 and CS-1.8.1-3 that were generated for UCAs issued by the human pilot.

3. Even if the ASEC does not receive a confirmation from the human pilot, the human pilot may be delayed in providing a confirmation or a modification to the flight path to the ASEC (e.g. due to the need for coordination, it takes the human pilot some time to understand the ASEC's proposed flight path). If the ASEC has a timeout programmed into it, it may wrongly decide that when the timeout has expired, it should simply execute its incorrect proposal without waiting for the human pilot.
4. The ASEC might have the correct process model of the state of the environment, the aircraft and the airspace around the aircraft and therefore plans a correct trajectory that does not pilot the aircraft toward another aircraft or object. However, when the ASEC coordinates with the human pilot, it receives an erroneous modification of the trajectory by the human pilot that does pilot the aircraft toward another aircraft or object. However, the ASEC incorrectly assumes that the human-modified path is correct so does not question it and executes it even though it flies toward another aircraft or object.
5. The ASEC might currently be on a trajectory that would pilot it toward another aircraft or object. The ASEC recognizes that its current trajectory must be modified, correctly computes a modified trajectory that avoids the other aircraft or object and then coordinates with the human pilot to confirm its new plan. However, the ASEC is programmed to continue its current flight path until the human pilot responds to its proposed new trajectory. If the human pilot is delayed in responding, the ASEC therefore continues to fly its existing trajectory toward the other aircraft or object.

For the other three basic scenario types (unsafe feedback path, unsafe control path and unsafe controlled process behavior), these scenarios are the same as those generated in the original STPA analysis in Section 3.4 because the ASEC is positioned at the same location in the control structure as the Piloting Controller was. So, those loss scenarios will not be replicated here.

STPA Analysis of Human Pilot Control Actions

Table 56: Example UCAs for Human Pilot Control Actions

Control Action	Providing	Not Providing	Provide too early/too late	Applied too long/Stopped too soon
Confirm Flight Path	UCA-3.1: Pilot confirms flight path when that flight path pilots the aircraft into another aircraft or object UCA-3.2: Pilot confirms flight	UCA-3.4: Pilot does not confirm flight path when a new flight path is needed to avoid piloting the aircraft into another aircraft or object	UCA-3.6: Pilot confirms the flight path too late after the last opportunity to avoid violation of minimum separation has passed	N/A

	path when that flight path is infeasible or outside the capabilities of the airframe UCA-3.3: Pilot confirms flight path that exceeds the bounds of the mission parameters (e.g. exceeds max altitude or operational area)	UCA-3.5: Pilot does not confirm the flight path when a new flight path is needed because the existing flight path no longer meets the needs of the mission (e.g. mission parameters change)	UCA-3.7: Pilot confirms the flight path too early before other personnel or cargo on board the aircraft are prepared for the change in flight path	
Modify Flight Path	UCA-3.8: Pilot provides modified flight path when the modified flight path will pilot the aircraft toward the other aircraft or object UCA-3.9: Pilot provides modified flight path that is infeasible or exceeds the capabilities of the airframe UCA-3.10: Pilot provides modified flight path that exceeds the	UCA-3.11: Pilot does not provide modified flight path when the path needs to be modified to prevent violation of minimum separation UCA-3.12: Pilot does not provide modified flight path when the path needs to be modified due to a change in the mission parameters	UCA-3.13: Pilot provides modified flight path too late after the aircraft begins to execute the unmodified new flight path that will result in a violation of minimum separation UCA-3.14: Pilot provides modified flight path too late to avoid the imminent violation of minimum separation caused by the	N/A

	bounds of the mission parameters (e.g. exceeds max altitude or operational area)		unmodified flight path	

Each of these UCAs can then be analyzed to identify the loss scenarios that could lead to these UCAs. Some examples are shown below for UCA-1 and UCA-8.

UCA-3.1: Pilot confirms flight path when that flight path pilots the aircraft into another aircraft or object

CS-G-3.1.1: Pilot confirms the flight path despite receiving feedback that the flight path pilots the aircraft into another aircraft or object. This might occur if:

1. Under a period of high workload or stress, the human pilot does not update their mental model of the state of the airspace (e.g. due to limited attention resources or cognitive tunneling) and uses their inaccurate mental model to evaluate the flight path instead. As a result, the human pilot does not realize that the flight path proposed by the ASEC pilots the aircraft into another aircraft or object and confirms it, wrongly believing that the flight path does not pilot the aircraft toward another aircraft or object.
2. Even though the human pilot recognizes that the flight path pilots the aircraft into another aircraft or object, they wrongly believe the other aircraft or object either is not actually present (i.e. a false positive detection) or will not actually be a collision threat in the future. As a result, the human pilot chooses to ignore the feedback indicating that the flight path proposed by the ASEC will pilot the aircraft into another aircraft or object and confirms the flight path anyway.
3. As a result of experience and operational adaptation, the human pilot might overly trust the ASEC to propose suitable flight paths and might become reliant on the ASEC. Such biases might therefore cause the human pilot to either assume that the ASEC is always right and confirm the flight path without checking it (i.e. essentially rubberstamping the ASEC's proposals) or perform only minimal checks to save cognitive effort. As a result, the human pilot does not notice that the flight path pilots the aircraft into another object or aircraft and confirms the flight path

CS-G-3.1.2: The human pilot receives feedback that does not indicate that the flight path pilots the aircraft into another aircraft or object. This might occur if:

1. The ASEC might not be able to detect the other aircraft or object for reasons discussed in CS-1.2.2. The feedback received by the human pilot from the ASEC therefore does not show that the flight path pilots the aircraft into another aircraft or object. Furthermore, DVE conditions might prevent the human pilot from independently receiving any other feedback besides the detections made by the ASEC. As a result, the human pilot is unable to

recognize that it is wrong because the human pilot can only observe what the ASEC can detect. This leads the human pilot to make the same incorrect decision that the ASEC makes, deciding that the flight path does not pilot the aircraft toward another aircraft or object. The human pilot therefore confirms that flight path.
2. Even if the ASEC might have detected the other aircraft or object in some but not all of its sensors, the human pilot might not be presented with sufficient feedback about the detections from individual sensors because the system may only show the pilot the post-processed integrated view of that data instead. As a result, they will be unable to notice that some sensors do detect an aircraft or object in the path proposed by the ASEC.

CS-G-3.1.3: The human pilot does not confirm the flight path but a confirmation is received by the ASEC. This might occur if:
1. An approval for a previously proposed flight path might be delayed in arriving. If the ASEC does not have a way to match a received approval to the flight path that the human pilot was approving, it may assume that any received approval is for the most recently proposed flight path. This asynchronicity might therefore result in approvals arriving out of order compared to the order in which flight paths are proposed. As a result, the ASEC wrongly believes the human pilot had approved the most recently proposed flight path even though the human pilot has not actually approved it yet.
2. If a controller in the system is compromised/hacked, an approval from the human pilot may be maliciously spoofed as originating from the human pilot even though they have not yet actually provided any confirmation.

CS-G-3.1.4: The human pilot does not confirm the flight path and no confirmation is received by the ASEC but the ASEC still proceeds to execute the new flight path and violates minimum separation. This might occur if:
1. A timeout might be programmed into the ASEC such that if a confirmation is not received from the human pilot within a threshold amount of time, the ASEC assumes its proposal is acceptable and begins executing it. As a result, if the human pilot is delayed because they need to coordinate with the ASEC or are unaware of the timeout time, they may take longer than the ASEC expects to make a decision and the ASEC may begin executing the flight path anyway. This is similar to CS-1.2.1-3.

UCA-3.8: Pilot provides modified flight path when the modified flight path will pilot the aircraft toward another aircraft or object

CS-G-3.8.1: The human pilot provides a modified flight path despite receiving feedback that the modified flight path will pilot the aircraft toward another aircraft or object. This might occur if:
1. Under conditions of high workload or stress, the human pilot might only pay attention to a limited amount of feedback and therefore not notice the feedback indicating that the modified flight path that they are proposing will pilot the aircraft toward another aircraft or object. As a result, they wrongly believe they are providing a suitable modified flight path.

2. When perceiving the feedback, the human pilot may apply unsafe biases or heuristics in using that feedback to update their mental model of the state of the airspace. For example, the human pilot may only notice the most salient feedback and miss less salient feedback that would show the presence of another aircraft or object. Alternatively, feedback received by the human pilot earlier in time that does not show the presence of another aircraft or object may be weighted more heavily in their decision-making when they propose a modified flight path compared to feedback received later. As a result, they do not use the later feedback effectively to recognize that the modified flight path they are providing will pilot the aircraft toward another aircraft or object.
3. Even if the human pilot correctly perceives the feedback, under conditions of high workload or stress or when they need to make a correction quickly, they might only consider a limited number of alternative flight paths or only consider the most available flight path options based on past experience. Alternatively, due to cognitive tunneling, the human pilot might become so focused on the immediate several seconds of the flight path such that they do not fully evaluate the rest of the flight path that they are proposing. As a result of any of these decision making heuristics and biases, they make a rushed decision when providing a modified flight path without fully considering whether that flight path avoids all other aircraft and objects. They therefore do not notice that they missed a location where there is another aircraft or object in the flight path they have chosen. As a result, they provide this modified flight path, wrongly believing that it does not pilot the aircraft toward another aircraft or object.

CS-G-3.8.2: The human pilot does not receive feedback that the modified flight path will pilot the aircraft toward another aircraft or object. These scenarios are the same as CS-1.1.2 above.

CS-G-3.8.3: The human pilot does not provide a modified flight path that pilots the aircraft toward another aircraft or object but the ASEC receives a modified flight path that does pilot the aircraft toward another aircraft or object. This might occur because:
1. A modified flight path might be delayed in arriving. If the ASEC does not have a way to match a received flight path to the flight path that the human pilot was reviewing, it may assume that any received flight path modification is for the most recently proposed flight path. This asynchronicity might therefore result in modified flight paths arriving out of order compared to the order in which flight paths are proposed. As a result, the ASEC wrongly believes the human pilot has provided a modified flight path that it should execute. If the state of the airspace has changed such that this previously provided flight path now intersects the location of another aircraft or object (even if it did not when it was originally provided), the ASEC will therefore have received a flight path modification that pilots the aircraft toward another aircraft or object even though the human pilot did not intentionally provide it.
2. If a controller in the system is compromised/hacked, a modified flight path that pilots the aircraft toward another aircraft or object is maliciously spoofed such that it appears to be sent by the human pilot even though the human pilot did not actually provide it.

CS-G-3.8.4: The human pilot does not provide a modified flight path that pilots the aircraft toward another aircraft or object and the ASEC does not receive one. However, the ASEC executes a modified flight path that does pilot the aircraft toward another aircraft or object. These scenarios are the same as CS-1.1.4 above.