SAFETY IMPLICATIONS OF AUTONOMOUS VEHICLES – SYSTEM THEORETIC PROCESS ANALYSIS APPLIED TO A NEURAL NETWORK-CONTROLLED AIRCRAFT

Ryan Bowers, Flight Test Engineer, 40th Flight Test Squadron

Dr. John Thomas, Massachusetts Institute of Technology

1. ABSTRACT

This paper investigates the safety implications of flight testing an Uncrewed Air Vehicle (UAV) controlled by a neural network-based flight autonomy software, and the utility of System Theoretic Process Analysis (STPA) in identifying risks. The host UAV in this case study includes various control regimes and handoffs over the course of a sortie including human control, traditional autopilot, and an artificial intelligence autonomy software trained using Deep Reinforcement Learning (DRL) machine learning techniques. The flight test operational environment includes flight in both civil and restricted airspace, and at least one nearby crewed chase aircraft to observe the UAV in flight. STPA was applied after traditional airworthiness and safety assessment processes but before flight test to identify and mitigate potential new hazards associated with the UAV technology and its operation. STPA was found to identify new risks, vulnerabilities, and undocumented assumptions that were used to create practical improvements in the technology, operational planning, and flight test. STPA produced additional mitigations related to the UAV, the automated run-time assurance mechanisms, human controls and other interactions with the UAV, and the overall operation of the autonomy. This paper summarizes some of the additional critical findings discovered by STPA prior to flight test, including:

1. The autonomy command limiters would not prevent unsafe combinations of control inputs that are individually within limits. Once STPA identified this gap, new mitigations were created to address it.

2. The original UAV safety mechanisms could not be easily modified to enforce the cleared envelope, so a new envelope protection mechanism was needed.

3. The original design of the human/autonomy handoff introduced potentially catastrophic scenarios related to confusion over whether the human pilot or the autonomy was in control of the host UAV. New mitigations were proposed to address these scenarios.

4. The UAV possessed a scripted maneuver designed to safely transition the UAV from autonomy control to human control, but that maneuver introduced new unforeseen risks.

2. INTRODUCTION

Today, new technologies like advanced autonomy and machine learning control systems are pushing the limits and challenging our ability to ensure safety during design, flight testing, and operations. Whether intended or not, a flight test of an autonomous vehicle or algorithm serves also as a test of the effectiveness of the methods used to develop that system, including the analysis and mitigation of hazards and the safety and airworthiness assessments of that system. In a perfect world with perfect methods, a system that reaches the flight testing stage might be

expected to already have completed the necessary analysis to ensure that all engineering assumptions have been documented and validated, that all potential hazards have been analyzed and mitigated effectively, and that the safety and airworthiness of the aircraft have already been proven by the standard airworthiness and safety assessments. But we do not live in a perfect world, and every flight tester knows to challenge these expectations. When catastrophic design assumptions or new unmitigated hazards are discovered during flight test planning or during actual flight tests, we should not only use this information to improve flight testing. We should use this information to improve the design of the aircraft, the way we design future aircraft, and the upstream methods we rely on including aircraft development, safety assessment, and airworthiness.

One general approach to autonomous vehicle development is to separate the development of the autonomy software from its host vehicle, for example through different engineering teams or even different companies, and then to integrate them after each has been independently matured. This has many advantages, such as the ability to mix and match autonomy software and host vehicles, faster development timeline, and better management of resources and expertise. However, it can also be a significant source of risk that may arise from complex interactions between the specific autonomy and host vehicle. For example, the autonomy engineers may assume that the host vehicle will mitigate certain hazards related to human/automation handoffs. Analysis and simulations may be performed prior to flight testing, but with the complexity of modern systems it is impossible to guarantee that every possible scenario is exhaustively studied.

These complex systems may also require updated safety analysis methods that are designed to handle that complexity. STPA is one such method, and an upcoming flight test of an experimental flight autonomy software on a host UAV presented an opportunity to examine STPA's utility for safety analysis of autonomous vehicles. This paper summarizes the findings from STPA and other methods.

3. SYSTEM DESCRIPTION

The subject of this paper is a combined system under test (SUT) consisting of an artificial intelligence agent (referred to as the autonomy) and a host UAV that is controlled by the autonomy in flight.

3.1 The host UAV

The host UAV was developed under a separate line of effort with the goal of producing a low-cost collaborative UAV that could accept a wide variety of autonomy systems.

The host UAV flies under three primary control modes over the course of a sortie:

- Control mode #1: Remotely piloted by a human remote pilot on the ground (Human "in the loop").
- Control mode #2: Piloted by an autopilot, where the human remote pilot may issue commands to the autopilot (Human "on the loop")
- Control mode #3: Flown by the autonomy to perform one task or a sequence of tasks.

Control modes #1 and #2 are referred to as the "native" control modes as they are part of the original design of the host UAV. By contrast, the autonomy is not native to the host UAV; the

autonomy and host UAV interact using an interface that was specifically developed for the two systems after the autonomy and host UAV were developed independently from one another.

3.2 The Autonomy

The autonomy under test for this project was a collection of neural networks that are trained using deep-reinforcement-learning (DRL) techniques in a simulation environment [1]. The autonomy's mission set includes basic aviation/navigation, flying in formation, and maneuvering in relation to beyond-visual-range entities. After simulation training, the autonomy was matured by integrating it onto a crewed aircraft. On this crewed aircraft, the pilot selected tasks for the autonomy to perform and disabled the autonomy if it exhibited potentially unsafe behavior. This demonstrated that the autonomy could control a real aircraft, and also characterized differences between the simulation and real world ("Sim-to-real transfer") that could be incorporated in training to improve future versions of the autonomy.

After maturation of the autonomy on the crewed aircraft, the autonomy was adapted to the host UAV. This involved four main phases:

1. A simulation model of the host UAV was developed for the autonomy to fly during training. This simulation model included aerodynamic properties and representations of the host UAV's envelope protection mechanisms (discussed in the safety enforcement mechanisms section) and internal flight computer logic.

2. The autonomy was retrained in simulation to fly the host UAV. The level of retraining required for transfer depends on the level of difference between the original and new airframe. If differences are relatively small, retraining may keep some of the previous training and only retrain for the differences in aircraft dynamics and envelope limits. However, the autonomy may need to be retrained almost entirely if the two airframes are very different.

3. The autonomy was tested in a hardware-in-the-loop (HITL) ground test where it interfaced with the host UAV's real flight computer and other flight hardware.

4. Finally, the autonomy was integrated onto the host UAV for live flight test in the air.

3.1.1 Autonomy done conditions

The autonomy's simulation framework includes "done" conditions that are used to end an episode of training or a simulation run. Different done conditions are implemented for success (the autonomy accomplished its goal), failure (the autonomy failed to achieve its goal within the required amount of time), or safety limit violation. These done conditions can also be used in live flight test to automatically terminate the autonomy when it finishes a task or, importantly, if it violates a safety limit (see safety enforcement mechanisms section later in this paper).

3.1.2 Live, virtual, constructive environment

The autonomy interacted with simulated aircraft within a Live, Virtual, Constructive (LVC) environment. An LVC environment allows real and simulated entities to interact as if they existed within one environment. "Live" refers to aircraft flying in the real world, "virtual" refers to a simulated aircraft piloted by a human in a ground simulator, and "constructive" refers to an aircraft or other entity that exists only in simulation.

For this project, the live entities included the autonomy-controlled host UAV as well as a safety chase aircraft that observed the host UAV from a safe distance. The autonomy interacted with constructive entities that it perceived as existing in the real world; in reality, they only existed in simulation. The autonomy's mission set included tasks that required it to fly close to other aircraft. However, because those aircraft were constructive entities, this did not introduce a risk of mid-air collision. LVC is not a focus of this paper but is useful in reducing risk that would be inherent to certain types of testing, such as those that involve other aircraft in close proximity.

3.3 Integration between the autonomy and the host UAV

3.3.1 Autonomy control implementation

In this project, the autonomy controlled the host UAV by providing three primary control inputs directly to the FC: vertical load factor (Nz) to control pitch, bank angle rate ($\dot{\phi}$) to control roll, and Throttle Lever Angle (TLA) to control longitudinal acceleration. Note that this control regime is not the only way that an autonomous algorithm could control a UAV, and other regimes have benefits and drawbacks. With respect to autonomy control terminology where "high-level" is providing control inputs to the airframe's full native control hierarchy and "low-level" is providing inputs directly to the airframe's control surfaces (or similar), this could be considered "mid-level" autonomy control.

The native control modes flow through the UAV autopilot (the remote pilot controls the UAV by changing the autopilot's target conditions). The UAV and autopilot were originally designed with the expectation that flight controls would always flow through the autopilot. This assumption allowed some aspects of the host UAV's safety mechanisms to be simplified because the autopilot flies in a relatively benign manner. However, the autonomy under test for this program was implemented in a way such that it bypasses the autopilot and provides input commands directly to the UAV flight computer (FC).

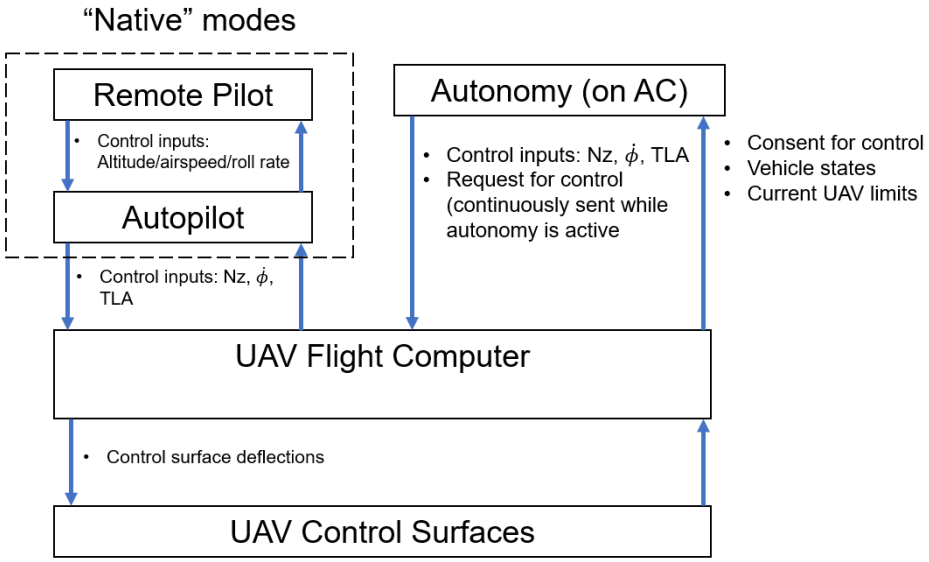

Figure 1. Control Structure showing multiple layers of UAV control.

3.3.2 Physical integration of the autonomy

The autonomy is hosted on an "autonomy computer" (AC) that was integrated into the host UAV to accommodate autonomy software. The AC interfaces with the UAV's FC to:

1. Orchestrate control handoff between the remote pilot and the autonomy.

2. Allow the autonomy to provide control inputs to the host UAV.

3. Allow the FC to send current UAV state information (Airspeed, altitude, attitude states and rates, GPS location, 3-axis accelerations, and constantly-updating UAV envelope limits) to the autonomy.

3.3.3 Transition between human and autonomy control

Handoff of control between the remote pilot and the autonomy is implemented as a two-way "handshake" where the autonomy must request control and the UAV FC must consent to control. Both the request and the consent are sent continuously in each message exchanged between the autonomy and the UAV; UAV control will be returned to the remote pilot if either the request or consent stop being sent.

To initiate the handoff process, the person managing the autonomy (this person can be the same person as the remote pilot, or could be a different person) selects a task for the autonomy to perform, which initializes the autonomy and allows it to begin sending control inputs and a "request for control" message to the host UAV flight computer. After the autonomy begins requesting control, the remote pilot enables "autonomy mode" on the host UAV. If the list of required conditions for handoff (not enumerated in this text) are met, then the autonomy's "request for control" message is received by the flight computer and the autonomy begins controlling the UAV.

If the autonomy does not request control within a pre-set timeout interval, autonomy mode will automatically disable and return control to the remote pilot. Termination of autonomy mode for any reason triggers a scripted transition maneuver that sets roll angle and pitch angle to zero and maintains the current airspeed before returning control to the remote pilot.

3.3.4 Safety enforcement mechanisms

Due to the experimental nature of the autonomy under test, and the unpredictability of DRL autonomy in general, mechanisms were implemented to ensure safety when the autonomy was controlling the UAV.

Training the autonomy to operate within the host UAV's flight envelope (airspeed, altitude, attitude, attitude rates, AoA and sideslip, and body axis accelerations) would, in an ideal world, prevent the autonomy from violating the flight envelope. However, the possibility of sim-to-real differences and other unexpected behavior means that this cannot be relied upon. Therefore, it was assumed while designing the UAV-autonomy integration that the autonomy would attempt to exceed the host UAV's flight envelope. Independent safety mechanisms were implemented to ensure that the UAV does not enter an unsafe state even if the autonomy attempts to do so. Three layers of safety enforcement mechanisms were implemented:

<u>UAV-enforced safety mechanisms:</u>

- Command limiters: the FC clips incoming autonomy commands (Nz, $\dot{\phi}$, TLA) at their maximum value.
- Inner loop control law safeties: The FC's "inner loop" control law includes features that prevent violation of certain limits like Nx, Ny, sideslip, etc.

- Automatic termination: The FC automatically terminates autonomy control and returns control to the remote pilot if the airspeed minimum, airspeed maximum, or altitude minimum are exceeded.

Autonomy-enforced safety mechanisms:

- Agent training: the agents that comprise the overall autonomy system are trained in simulation to stay within a specified set of flight conditions.
- Autonomy "done" conditions: Done conditions can be set on any flight parameter to act as an additional safety measure. If the autonomy exceeds any of these parameters, the done condition will trigger and the autonomy will terminate itself.

Test procedures for manual termination: Even if other safety mechanisms fail, the remote pilot is able to terminate the autonomy at any time and regain control of the UAV. The test team developed test limits for all flight envelope parameters, such that the remote pilot would immediately terminate the autonomy if any test limit was violated.

Because the autonomy is the system under test, the autonomy-enforced mechanisms are considered to be less robust than other methods. The UAV-enforced safety mechanisms and test procedures were designed such that even if the autonomy-enforced mechanisms failed, the host UAV would never enter an unsafe state.

4. CHALLENGES

The design described above presents several unique challenges for safe flight operation. The development and safety analysis attempted to investigate and mitigate each of these challenges. The most interesting challenges include:

1. Flight testing machine learning-based autonomy is itself a unique challenge. Trained neural networks are inherently non-deterministic and unpredictable. Undesirable behavior can be discouraged using the reward function but cannot be discretely disallowed in the same way as a rule-based autopilot. For example, the autonomy can be strongly discouraged from violating an airspace boundary by providing a large negative reward, but this is not the same as directly programming it to stay within that boundary – there is always the possibility that the autonomy will choose to violate the boundary and accept the negative reward in pursuit of a more valuable reward.

2. In addition to the inherent complexities of flight testing autonomy, autonomy development is currently in its infancy and flight testers cannot currently rely on decades of experience and best practices for these systems

3. The paradigm of integrating an autonomy system and a host UAV that had been developed separately has advantages but is also a significant opportunity for risks involving the interface, incompatibilities, and unexpected interactions.

4. The host UAV's specified and designed flight envelope had not been fully cleared prior to autonomy flight testing. The test organization decided to restrict the autonomy to only fly within the cleared envelope.

5. One philosophy for safe autonomy testing that is gaining traction is to robustly validate the independent safety mechanisms present on the host aircraft, which, if designed properly, would

negate any possible risk posed by the autonomy itself. However, in this project, the UAV's safety mechanisms had not been validated in flight. As a result, flight test procedures for this project were very restrictive in order to further mitigate safety risk. Future autonomy testing may be much less restrictive given a robustly validated set of aircraft safety mechanisms.

5. SYSTEM THEORETIC PROCESS ANALYSIS

5.1 Overview of STPA

STPA [2] is a method to identify potential flaws and vulnerabilities in a system that can lead to losses like accidents, mishaps, mission losses, or other unacceptable losses. STPA's strengths include the ability to analyze:

- Systems with complex autonomy
- Systems with complex operational environments, non-trivial or unexpected human/automation interactions
- New system behaviors that may emerge from complex component interactions
- Human factors such as system-induced human confusion
- Intended system functions that may inadvertently lead to hazards with or without a failure.

STPA is commonly used for safety-driven design, test, and assurance in civil aviation, military aviation, automotive, space, and other industries. Previous projects have applied STPA to autonomous vehicles [3] [4] [5].

STPA was chosen for this project because it has been demonstrated as an effective safety analysis method for the types of challenges involved in this project: systems with complex autonomy, systems that depend on and influence complex human interactions, systems that have many subsystem interactions that are not well understood, and new systems that do not have historical data or a large set of past experiences for a safety analysis to draw from.

In this project, the DRL-trained autonomy is modeled as a "black box" to limit the assumptions made about the final detailed software design and the behaviors that may or may not be expected from a neural network-based agent. Instead, the focus of this effort is on identifying what hazardous behaviors may not be effectively mitigated by the technology and the operational environment, how the technology and the operational environment can be improved, and what hazardous interactions between the autonomy, the host UAV, the remote operators, the chase aircraft, and other aspects of the operational environment must be mitigated to ensure safe flight test and operation. Special emphasis is placed on using STPA to identify undocumented engineering and human factors assumptions that may have been previously overlooked and must be validated or mitigated.

The STPA findings included general hazards that had previously been identified as well as new hazards, causes, and discoveries that were not previously identified by the team. The following STPA results include each of these cases in order to provide a common reference with as many lessons as possible for future projects.

5.2. Hazards and losses

<u>The following losses were defined for the STPA analysis:</u>

L-1: Loss of life.

L-2: Loss or damage to host UAV or its environment.

L-3: Loss of program timeline.

L-4: Loss of credibility or reputation.

L-5: Loss of future test capability.

L-6: Inability to collect test data.

The following hazards were defined:

Table I. STPA hazards and connections to losses.

#	Hazard	Connection to losses
H-1	Host UAV too close to ground/objects	[L-1] – [L-6]
H-2	Host UAV too close to other aircraft	[L-1] – [L-6]
H-3	Host UAV departs authorized test area	[L-1] – [L-6]
H-4	Host UAV lands outside authorized landing area	[L-1] – [L-6]
H-5	Host UAV exceeds operational envelope	[L-1] – [L-6]
H-6	Host UAV unable to provide useful test data	L-3, L-5, L-6
H-7	Host UAV becomes uncontrollable	[L-1] – [L-6]

STPA was applied to typical safety-related losses such as L-1 and L-2 as well as broader losses like L-6: Inability to collect test data. L-6 covers scenarios in which the autonomy's behavior triggers an existing safety mechanism. Even though the safety mechanism keeps the system safe, it may be impossible to test the autonomy's performance if it frequently performs actions that result in automatic termination. Loss L-6 enabled the team to mitigate test efficiency risks as well as safety risks.

STPA was also found to be applicable to L-5: Loss of future test capability and L-4: Loss of credibility or reputation. Autonomy development programs, especially for military applications, can be polarizing and subject to increased scrutiny and risk of cancellation. The STPA process acknowledges that a hazard can cause more than one loss, or potentially no losses depending on best-case and worst-case environments. Some hazards (e.g. near-collision or airspace violation) could result in program cancellation or other indirect but significant losses even if L-1 and L-2 are prevented.

5.3 Control Structure

The primary STPA control structure developed for the analysis is shown below. It includes the following:

- The DRL-trained Autonomy
- The host UAV with the FC and control surfaces
- The primary human actors during flight test:
    - **Test conductor** – Manages flight test execution flow.

  - **Human remote pilot** – Flies the UAV between autonomy test points, during emergencies, and during any other off-nominal situation. Enables and disables autonomy mode.
  - **Autonomy operator** – Enables and disables the autonomy
  - **Telemetry engineers** – Observe UAV telemetry for safety of flight and communicate primarily with the test conductor
  - **Chase pilot –** Communicates with control room team to provide additional Situational Awareness (SA) of the UAV (e.g. if datalink is lost), help deconflict with weather and other aircraft (the control room does not have visual SA of the UAV) and inspect the UAV for structural damage after anomalies. In time-sensitive situations, the chase pilot can talk directly to the human pilot to provide direction.
- **UAV Safety Mechanisms –** the mechanisms that automatically terminate the autonomy and limit its input commands, as described in section 3.3.4.

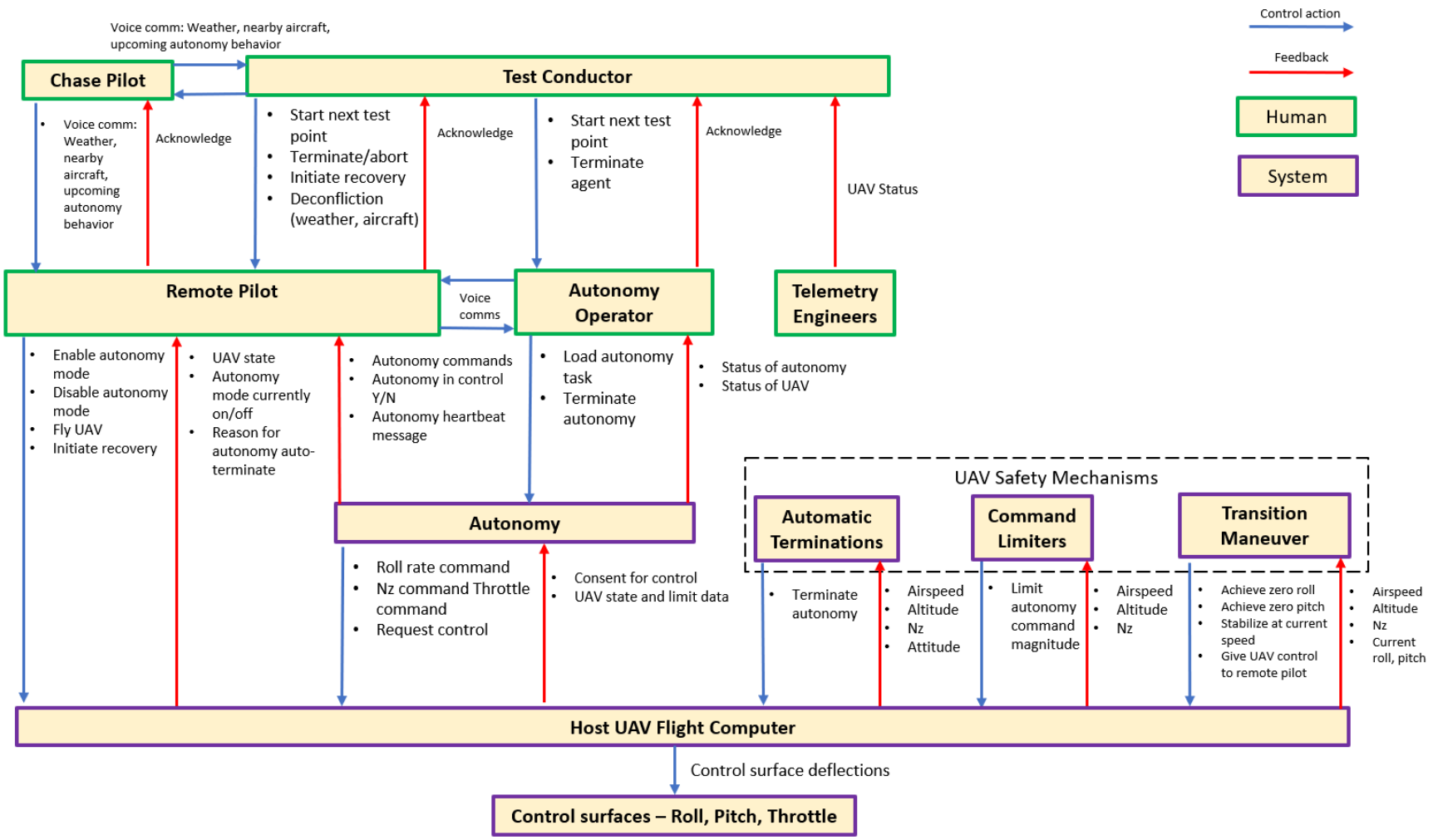

Figure 2. The primary STPA control structure built for the autonomy, host UAV, and test conduct team.

Other control structures at different levels of detail were created as required to support more detailed analysis. A control structure for the inner workings of the autonomy was out of scope for this initial effort.

After developing the control structure, the team identified unsafe control actions (UCAs): control actions that are intentionally identified as possible in the control structure and are not necessarily unsafe all the time but become unsafe in certain situations. Examples of UCAs that were identified are shown in the table below:

Table II. UCA Examples

Unsafe Control Action	Associated Hazards
<Autonomy> provides oscillatory Nz or $\dot{\phi}$ control inputs to the <UAV>	H-1 – H-7
<Auto State Trips> do not terminate <autonomy> when the autonomy exceeds the cleared flight envelope.	H-5 – H-7
<Human pilot> does not fly the host UAV when they are in control.	H-1 – H-4
<Host UAV Flight Computer> does not disable autonomy mode when autonomy is not sending commands	H-1 – H-4, H-7

After identifying UCAs, the team identified loss scenarios that could cause those UCAs to occur. The “Findings” section below describes the most noteworthy loss scenarios that were identified. Additional STPA results can be found in the Appendix.

6. FINDINGS

6.1 List of Findings

During the STPA sessions, around 50 new critical safety-related findings were produced which led to 49 updates to test procedures. Since a full description of every finding is not possible in the scope of this paper, a limited subset of the findings are summarized below:

1. The autonomy command limiters would not prevent unsafe combinations of control inputs that are individually within limits.

2. The UAV safety mechanisms could not be easily modified to enforce the cleared envelope, so a new envelope protection mechanism was needed.

3. The control handoff’s “handshake” design could make it difficult to know whether the human pilot or the autonomy was in control of the host UAV.

4. The UAV possessed a scripted maneuver designed to safely transition the UAV from autonomy control to human control, but that maneuver introduced new unforeseen risks.

6.2 Details of findings

6.2.1 Finding 1: Unsafe combinations of autonomy control inputs that could not be prevented by the control limiters.

During simulation validation of the UAV-enforced safety mechanisms, multiple situations were discovered in which the autonomy could place the host UAV in an unsafe state even while

adhering to all existing safety enforcement mechanisms. This was often due to combinations of control inputs that would not be dangerous alone but were dangerous when combined.

Table III. Loss Scenario 1.1.

#	UCA	Causes	Loss Scenario
LS-1.1	Autonomy provides oscillatory Nz or $\dot{\phi}$ control inputs to the UAV.	(Control algorithm) The command limiters only constrain command inputs to a min/max range. They do not account for previous or future states. (Control algorithm) Autonomy does not avoid control inputs that cause departure from controlled flight (or aerodynamics not accurately modeled)	The autonomy provides oscillatory command inputs (e.g., alternating between min and max Nz every 1 second). This causes the UAV to depart controlled flight and collide with terrain, property, another vehicle etc. [L-1 – L-6]

For example, modeling and simulation (M&S) predictions showed that rapid control reversals (e.g. alternatively commanding maximum and minimum Nz every 1 second) would cause the UAV to depart controlled flight at certain conditions, even if the commands were within limits.

Another M&S prediction showed that when the dynamic Nz limit was low (for example, at low airspeeds), the autonomy could command the UAV to enter a steep turn that could not be coordinated even with maximum Nz command. This resulted in a rapid descent until the low altitude limit was tripped and terminated the autonomy. Because the autonomy's command limiters applied to bank angle rate but not bank angle itself, there was no way to prevent this behavior using the command limiters. The UAV safety mechanisms can be used to mitigate this safety risk by triggering terminations, but excessive terminations represented a test efficiency risk.

Table IV. Loss Scenario 1.2.

#	UCA	Causes	Loss Scenario
LS-1.2	Autonomy does not provide sufficient positive Nz control during a turn.	(Control algorithm) Autonomy is not capable of providing sufficient positive Nz to coordinate the turn due to flight conditions (e.g. low airspeed) or UAV state (e.g. heavy weight). (Control algorithm) The autonomy's training does not prevent it from commanding a roll angle that cannot be coordinated.	The autonomy enters a steep bank, and the current Nz limit is below what would be required to maintain a coordinated turn. The UAV begins to descend and is terminated when the low altitude limit is tripped. If the autonomy does this frequently, it would be terminated constantly leading to a loss of test effectiveness/efficiency. [L-6]

6.2.3 Finding 2: The UAV safety mechanisms could not be modified to enforce the cleared envelope, so a new envelope protection mechanism was needed.

As stated in the challenges section, the test team chose to restrict the autonomy to fly within the current proven UAV flight envelope rather than the full advertised envelope. This posed a challenge because, with a few exceptions such as a heightened minimum altitude limit, the host UAV safety enforcement mechanisms enforced the full advertised UAV envelope values rather than the cleared envelope. This is because these mechanisms were implemented when the UAV was originally designed and it was expected that the remote pilot would avoid flying to unproven parts of the envelope.

The autonomy's training was meant to prevent it from flying to uncleared parts of the envelope. However, because the autonomy's training and done conditions had not yet been validated in live flight, the fact that the UAV safety mechanisms enforced the full envelope represented a gap in the safety enforcement system.

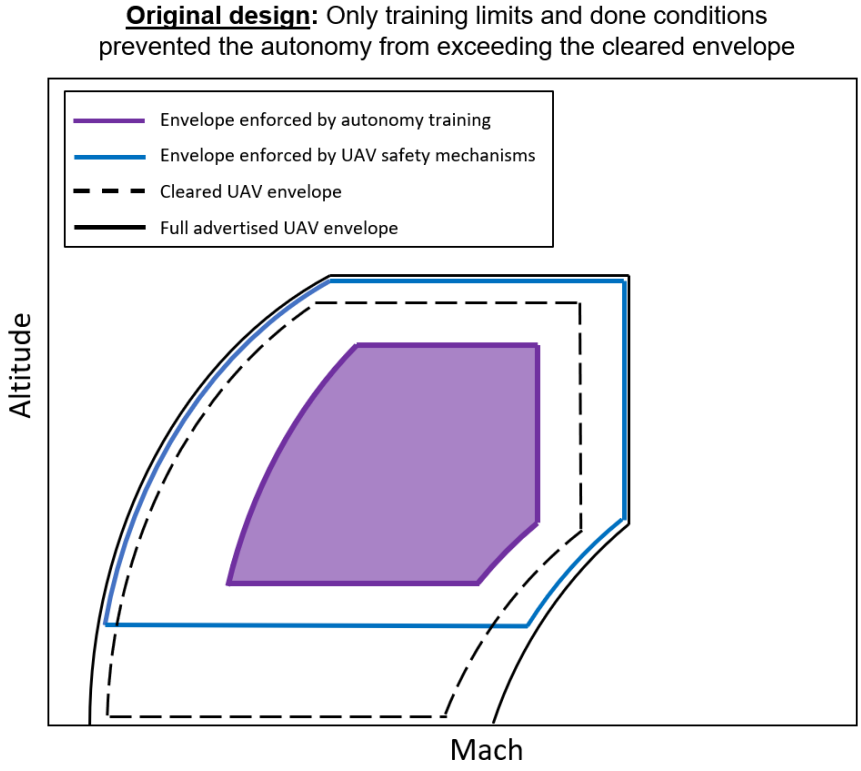

Figure 1. Notional Mach-Altitude envelope diagram.

Table V. Loss Scenario 2.1.

#	UCA	Causes	Loss Scenario
LS-2.1	UAV safety mechanisms do not terminate autonomy when it exceeds the	Control algorithm: Safety mechanisms are coded with the wrong envelope values for flight	UAV exceeds the cleared altitude/airspeed envelope but remains within the advertised envelope, and the UAV safety mechanisms do not trigger because they enforce the full advertised envelope. However, due to errors in predicting the full envelope, the UAV is

	cleared flight envelope.	test and cannot be changed.	unstable outside of the cleared envelope and departs controlled flight or experiences structural damage.

Besides the primary envelope limits of airspeed, altitude, and Nz, the other structural and aerodynamic limits (e.g. N_y, angle of attack) were enforced by inner-loop controllers that also enforced the full advertised envelope. This was not a concern in the host UAV's native control modes because the autopilot flies well within the limits. However, the autonomy could fly completely differently and potentially approach the advertised limits, and there was no real-world data to provide sufficient confidence in the UAV's behavior at these conditions.

Discussions with the host UAV design team revealed that the UAV's safety mechanisms could not be modified to enforce different values for the structural and aerodynamic limits, so a new safety enforcement mechanism would be required.

6.2.5 Finding 3: The control handoff's "handshake" design could make it difficult to know whether the human pilot or the autonomy was in control of the host UAV.

As explained in section 3.3.3, the process for exchanging control of the UAV between the human pilot and the autonomy involves a handshake mechanism where the remote pilot engages an "autonomy mode" switch on the ground terminal that allows the autonomy to begin requesting control of the UAV. Autonomy mode automatically disables after a preset time if it does not receive a request for control from the autonomy.

The ground terminal features a digital light that indicates that autonomy mode is enabled, which is the remote pilot's primary indication that the autonomy is in control of the UAV. However, autonomy mode does not explicitly indicate that the autonomy is in control of the UAV, only that it is *allowed* to control the UAV. This raises the possibility that the human pilot could believe the autonomy is in control because the autonomy mode light is on, but in reality, the autonomy is not sending commands and the autonomy mode is in the process of timing out. Depending on the timeout length, there could be a significant amount of time where no one is flying the host UAV because the human pilot thinks the autonomy is flying when it is not.

Table VI. Loss Scenario 3.1.

#	UCA	Causes	Loss Scenario
LS-3.1	Human pilot does not fly the host UAV when the human pilot is in control.	Cause 1 (Process model): Human pilot believes the autonomy is in control because the autonomy mode switch on the ground terminal is on.	Autonomy stops sending control commands. The autonomy mode logic is in the process of timing out, but by design this information is not provided to the human pilot. The human pilot would only see that autonomy mode is still enabled. As a result, the human pilot does not maneuver the UAV. In the worst case, a collision may occur [L-1 – L-6].

Another potential issue with the control handoff arises from the fact that autonomy mode is maintained by a "request for control" message from the autonomy, not by actual autonomy control inputs. For any number of reasons, it is possible that the autonomy could continue to send a request for control even if it is not sending actual input commands. As a result, the UAV FC would not disable autonomy mode even though the autonomy is not in control.

Table VII. Loss Scenario 3.2.

#	UCA	Causes	Loss Scenario
LS-3.2	UAV FC does not disable autonomy mode when autonomy is not sending commands	Cause 1 (Feedback): UAV FC is still receiving request for control but is not receiving any other commands.	The UAV stops receiving control inputs from the autonomy, but by design this does not trigger the FC to disable autonomy. The UAV FC considers the autonomy request for control as valid even if the control inputs from the autonomy disappear. As a result, no one is flying the UAV until it causes a hazardous aircraft state and the human pilot or an auto safety trip may attempt to terminate the autonomy. [L-6]

6.2.1 Finding 4: The scripted transition maneuver, designed to safely transition the UAV from autonomy control to human control, introduced new unforeseen risks.

When the scripted transition maneuver described in section 3.3.3 was investigated in more detail, scenarios were discovered in which the maneuver could cause unsafe situations and prevent the human pilot from regaining control. Because the transition maneuver waits until its target conditions (zero roll angle, zero pitch, and stable at current airspeed) before handing control back to the human pilot, it may take an excessive amount of time for the human to regain control if something prevents those conditions from being reached (e.g. turbulence or loss of control surface authority due to a malfunction). In this case, the transition maneuver may prevent the human pilot from responding to an urgent hazard such as a nearby aircraft that the UAV cannot sense. As a result, despite the transition maneuver being designed to safely transition from autonomy to human control, it could be the cause of hazards as well.

Table VIII. Loss Scenario 4.1.

#	UCA	Causes	Loss Scenario
LS-4.1	<Transition Maneuver> does not [transition control] to <human pilot> when the	Cause 1: Environmental conditions (e.g. turbulence) or UAV condition (e.g. mechanical failure) prevent the maneuver's target conditions from being reached.	Transition maneuver refuses to return control to the human pilot because it is unable to stabilize the aircraft and satisfy the exit criteria.

	autonomy is terminated.	Cause 2 (Feedback): UAV Flight Computer does not send feedback to the transition maneuver controller that its conditions have been met (due to sensor fault, etc.)	The human pilot is then unable to maneuver to avoid obstacles and other hazards, and the UAV may collide with an obstacle or must be emergency landed. [L-1 – L-6]

7. CONTRIBUTIONS AND IMPACTS

The STPA findings can be used to improve the technology and its operations by ensuring that mitigations are incorporated for each of the safety issues identified. This section summarizes the contributions and impacts related to the sample of findings in the previous section.

7.2 Solutions to Finding 1.

Because the safety of the autonomy had not yet been proven, the team identified two possible solutions for Finding 1 to prevent unsafe control inputs that were not adequately handled by the original safety mechanisms:

> Solution #1: Modify the safety mechanism algorithms to include time history of control inputs (to prevent oscillatory inputs that would cause loss of aerodynamic control) and the relationships between different control inputs (to prevent behavior like steep bank angles that could not be coordinated with the current N_z limit).
>
> Solution #2: Set dynamic autonomy done conditions to accomplish the same intent as solution #1.

The findings from STPA were flexible in the sense that more than one solution could be proposed to address the findings. This allowed the potential solutions to be ranked and prioritized based on other factors, such as the program scope, solution effectiveness, and other factors. The solutions above are included as examples only, not as a final or total solution for all possible causes. For example, the done conditions in Solution #2 had to be defined to target specific behaviors, so only "known" unsafe behavior could be prevented in that solution. Elsewhere, the team considered the possibility of other unsafe behavior not covered by the dynamic done conditions in Solution #2.

7.3 Solutions to Finding 2.

To prevent the autonomy from violating the cleared flight envelope, the test team defined "test limits" for each flight envelope parameter. If a test limit was violated, the remote pilot would terminate the autonomy and the autonomy done conditions would also trigger.

The placement of the test limits in relation to the autonomy training limits and the UAV's envelope was driven by the desired balance between safety and test efficiency. Sim-to-real differences could cause the autonomy to operate at its training limits or even exceed them, so setting the test limits equal to the training limits could lead to frequent autonomy terminations that prevented data collection. On the other hand, test limits that were too wide could lead to unintentional violation of

the cleared envelope due to the done condition being delayed (or failing to trigger) and the remote pilot's reaction time for termination.

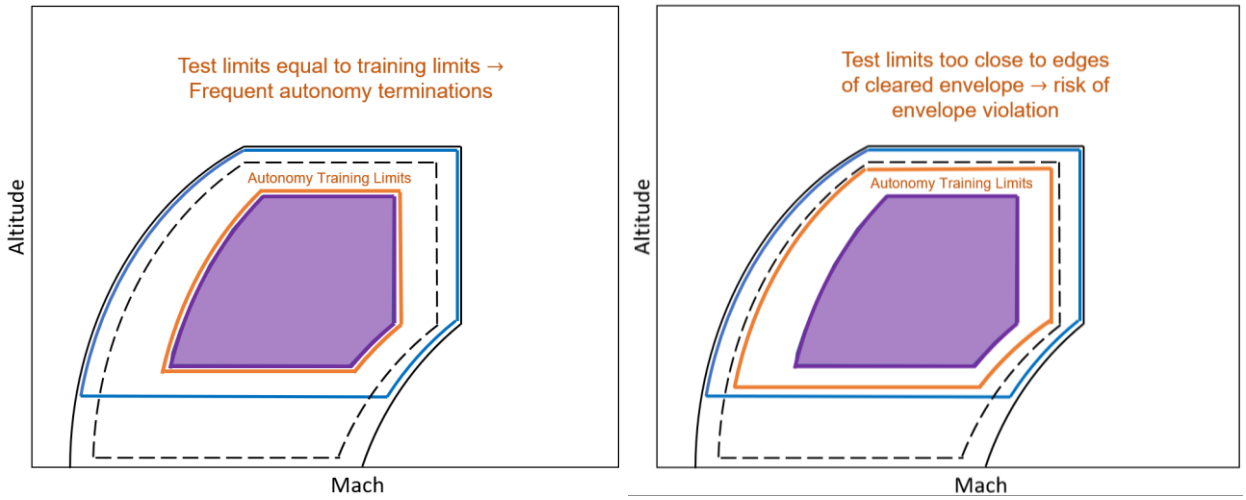

Figure 2. Placement of test limits – two extremes.

The test team found a balance between the two extremes by setting the test limits far enough outside the training limits that the autonomy could briefly exceed its training limits without being terminated, but far enough from the edges of the cleared envelope that the remote pilot had time to terminate the autonomy even if the test limit was exceeded.

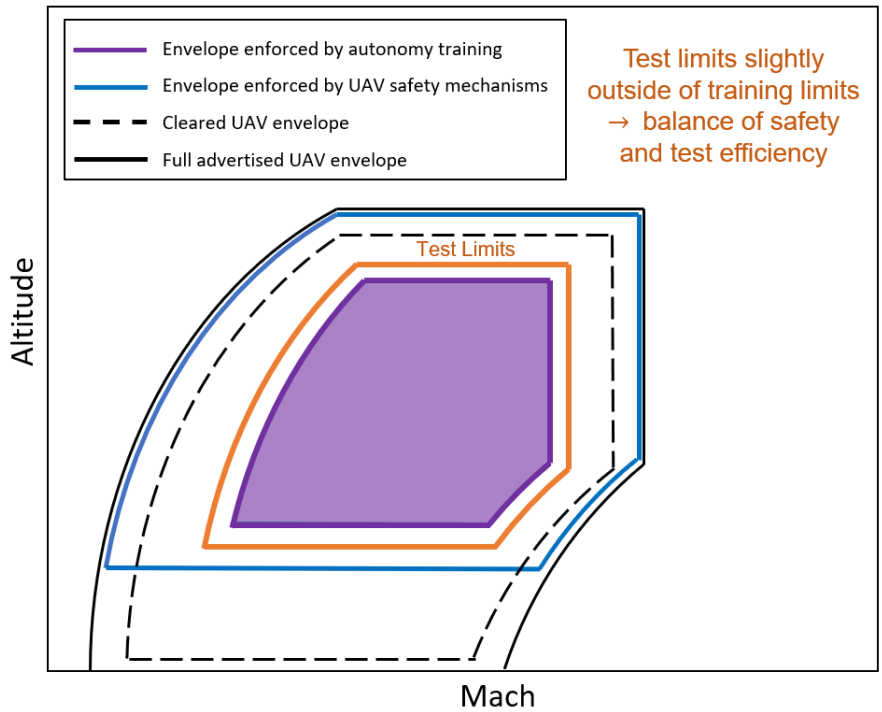

Figure 3. Test limits placed to balance safety and test efficiency.

7.5 Solutions to finding 3.

Several possible solutions were identified to clarify whether the autonomy was actively in control of the UAV:

Solution 1: Make the autonomy mode timeout very short (e.g., less than 3 seconds).

Solution 2: Show the autonomy mode timeout on the ground terminal so the remote pilot can see when the autonomy is not actually sending commands.

Both solutions can be considered along with their tradeoffs. For example, solution #1 requires good coordination between the remote pilot and the autonomy operator to time the handoff.

The possibility of the autonomy sending a request for control but not actually sending control inputs could be solved by adding a check in either the UAV FC or within the autonomy that terminates autonomy mode if control inputs are not being sent to the UAV.

7.1 Solutions to finding 4.

The scripted transition maneuver should be designed with an override option to allow the human pilot to regain control even if the maneuver's target conditions are not met. This risk may also be mitigated procedurally: the human pilot should be briefed on the possibility of a hung transition maneuver and instructed to terminate the autonomy early depending on potential hazards to allow for a greater lag before regaining control.

8. CONCLUSIONS

8.1 Discussion of the system's major design choices.

The safety analysis found specific safety concerns related to the design of the system under test. Some limited conclusions about the major design choices of the system are given below.

- **Integration of separately-developed host UAV and autonomy:** In general, integrating a separately-developed host UAV and autonomy means that some original design decisions in one of the two systems may no longer achieve their intended results, or the design of one system may violate assumptions in the other system. How can one systematically identify what these are? In this project, STPA was used to examine the unintended behaviors caused by the combined integration of the host UAV and autonomy. For example, Finding 1 found that although the UAV's native control modes appeared reasonable as originally designed, these modes were bypassed by the autonomy and several key safety mechanisms no longer existed.
- **Interface between host UAV and autonomy:** STPA also considered the interface between the autonomy and the host UAV. Because the interface was based on digital messages directly exchanged between the autonomy and the UAV FC with standard protocols and checks, the control path was found to minimize the possibility of control path errors. However, other interface concerns were identified (and mitigated) by considering the overall system behavior and not just control path transmission.
- **Reliance on safety mechanisms that are independent of the autonomy:** As described earlier, the independent safety mechanisms were too simple to prevent all unsafe autonomy behavior. This placed a greater responsibility for safety on the autonomy itself than originally desired and assumed. Additionally, the inner loop control law safety mechanisms introduced in section 3.3.4 were not directly analyzed, but they raised initial concerns about whether they

could adequately enforce safety for an autonomy system that did not fly in as benign a manner as the UAV's native autopilot. This type of safety mechanism deserves significantly more analysis, and future developers of platforms intended for use with various autonomous algorithms must ensure that the aircraft's inner loop safeties are sufficient for a wide variety of flying styles.

8.2 Observations about the use of STPA.

STPA was an effective framework for the safety analysis of this autonomous aircraft and its operation. The control structure was a helpful model for methodically identifying and analyzing the major components of the system under test, and the controller-controllee relationship at the heart of STPA proved to be a useful mindset even during early discussions before the specific STPA results were known.

Several STPA steps and discussions challenged whether the autonomy and its training could be assumed to be safe, and identified the additional safety mechanisms that were necessary to handle potential unsafe autonomy behavior. In this project, STPA treated the autonomy as a black box and did not model the internal operations within the autonomy. Previous projects [5] have demonstrated the value of applying STPA within the autonomy itself, and it would be valuable for future STPA efforts to model the autonomy in greater detail and analyze its internal decision making.

The team initially struggled when developing the control structure (section 6.1.2) to determine the appropriate hierarchy between components of the autonomy, the host UAV, and the independent safety mechanisms. For example, the autonomy provides control inputs to the UAV FC, which suggests that the autonomy should be above the UAV in the control structure hierarchy. However, the UAV FC must *allow* the autonomy to control the UAV and can retract that control at any time, which could suggest the opposite hierarchy to some who are new to STPA. Qualified STPA training and facilitation is recommended for those new to STPA to avoid these types of mistakes. Simply reading descriptions of STPA in papers and books is not enough to develop skill and proficiency in applying STPA.

The STPA results detailed in this paper required approximately 136 person-hours across three days. Although all participants had a basic understanding of the system before applying STPA and some participants were subject matter experts, approximately half of this time was dedicated to learning additional details about the system, answering new questions that STPA had raised about the system, and modeling the control structure in Figure 2 – something that had never been modeled before despite having a completed design ready for flight testing. At the end of this limited effort, the team agreed that more time could have been used to perform STPA and investigate additional concerns. Any future applications of STPA to a system of this complexity should account for the significant amount of time required to learn about the system and to explore the new concerns raised by STPA that have not been raised before.

The STPA control structure and other STPA steps were found to provide a common language for flight testers, pilots, software engineers, and other experts to communicate their understanding of the system and to identify the critical assumptions that may appear reasonable in one domain but are not reasonable from the perspective of another domain. For example, the primary feedback in Figure 2 from the autonomy to the human operator confirming that autonomy mode was enabled may appear reasonable from an engineering perspective but not from an operator

perspective, because the autonomy may be enabled but not ready (i.e., it may appear that the autonomy has taken control when the human operator is still in control). STPA's approach to systematically analyze the assumptions and interactions between physical layer, the automation and autonomy layers, and the human layers provided a powerful way for domain experts—including flight testers, software autonomy experts, and pilots--to communicate and identify undocumented assumptions that could not be easily identified by studying any one domain in isolation.

The team found that the STPA findings were valuable for improving flight test safety as they went beyond typical hazard analysis results. There is also an opportunity for the STPA findings to be produced much earlier during the development process to improve the design decisions as they are made rather than waiting until after the airworthiness and safety assessments had already been performed and a prototype is ready for flight testing. If STPA had been used earlier, the design changes and solutions needed to ensure safe testing and operation would have already been implemented and they would have been much less costly to implement at an earlier stage. There is also an opportunity for the STPA results to be reused by different entities, including the development team, the safety team, the flight test team, and others with much less effort.

8.3 Closing thoughts.

This paper summarizes some of the insights produced during a limited STPA safety analysis that primarily focused on areas of the system that were uncertain and expected to pose risks. The findings shared in this paper are not intended to be comprehensive, but instead to provide a starting point for future teams who may be considering STPA or may be wondering what benefits may have been found in similar projects. As the field of autonomous vehicle development matures, it is the hope of the authors that papers like this one will play a small role in the development of best practices and industry standards.

The views expressed in this paper are those of the author and do not necessarily reflect the official policy or position of the Department of the Air Force, the Department of Defense, or the U.S. government.

9. ACKNOWLEDGEMENTS

The authors would like to thank Ryan Lambert, Matthew Niemiec, Major Daniel Hayes, Major Ross Elder, Captain Tyler Brown, Nathan Cook, Byron "KJ" Johnson, and Lieutenant Elizabeth Pennington for their support of the STPA sessions that made this paper possible.

10. REFERENCES

[1] Francois-Lavet, V. et al., "An Introduction to Deep Reinforcement Learning", Foundations and Trends in Machine Learning: Vol 11, No. 3-4, 2018.

[2] Leveson, Thomas, "STPA Handbook", March 2018.

[3] Hobbs, K. et al., "Systems Theoretic Process Analysis of a Run Time Assured Neural Network Control System", 2022.

[4] Robertson, J., “Systems Theoretic Process Analysis Applied To Manned-Unmanned Teaming,” master’s thesis, Aeronautics and Astronautics Dept., Massachusetts Institute of Technology, January 2019.

[5] Schmid, M., “A Design Process and Certification Strategy for Autonomous Vehicles,” master’s thesis, Aeronautics and Astronautics Dept., Massachusetts Institute of Technology, June 2020.

11. BIOGRAPHY

Ryan Bowers

Ryan Bowers has been a flight test engineer at the 40th Flight Test Squadron at Eglin AFB since March 2021, and has spent most of that time supporting flight test planning and execution of autonomous group 5 UAVs and fighter aircraft. He graduated from Georgia Tech with a B.S. in Aerospace Engineering in 2020 and plans to pursue a graduate degree in Fall 2024.

Dr. John Thomas

Dr. John Thomas is co-director of the Massachusetts Institute of Technology’s Engineering Systems Laboratory. His research at MIT studies engineering mistakes and human errors that lead to accidents and other losses in order to identify gaps and improvements in modern engineering and safety assessment methods. His background includes fixed wing aircraft, rotary wing aircraft, manned and unmanned applications, and autonomous vehicles in air, space, and ground applications. He collaborates extensively with civil and DoD aircraft manufacturers, operators, regulators, and others. He teaches classes on system safety, system engineering, software engineering, and human factors. He currently serves on industry standards committees including the SAE S-18 committee responsible for standard aircraft development and safety assessment processes ARP4761A and ARP4754B.

APPENDIX. DETAILED STPA RESULTS

This appendix lists STPA UCAs and loss scenarios that support the major findings in section 6. A sample of additional STPA products selected for publication are also listed for reference. It is important to note that the full set of UCAs and scenarios cannot be included in this paper.

Table IX. STPA products for Finding 1.

#	UCA	Causes	Loss Scenario	Solution
LS-1.1	<Autonomy> provides oscillatory Nz or $\dot{\phi}$ control inputs to the <UAV>	(Control algorithm) The command limiters only constrain command inputs to a min/max range. They do not account for previous or future states. (Control algorithm) Autonomy is not trained to avoid control inputs that cause departure from controlled flight (or aerodynamics not accurately modeled)	The autonomy provides oscillatory command inputs (e.g. alternating between min and max Nz every 1 second). This causes the UAV to depart controlled flight and collide with terrain, property, another vehicle etc. [L-1 – L-6]	Add temporal knowledge to command limiter algorithm (e.g. define a maximum difference between current command and previous command). Use done conditions to terminate the autonomy if it begins to provide oscillatory control inputs.
LS-1.2	<Autonomy> does not provide sufficient positive N_Z control to the UAV during a turn.	(Control algorithm) Autonomy is not capable of providing sufficient positive Nz to coordinate the turn due to flight conditions (e.g. low airspeed) or UAV state (e.g. heavy weight). (Control algorithm) Autonomy training does not prevent it from commanding a roll angle that cannot be coordinated.	The autonomy enters a steep bank, and the current Nz limit is below what would be required to maintain a coordinated turn. The UAV begins to descend and is terminated when the low altitude limit is tripped. If the autonomy does this frequently, it would be terminated constantly leading to a loss of test effectiveness/efficiency. [L-3]	Train autonomy to limit its bank angle to the maximum that can be coordinated based on the current N_Z limit. Consider adding a 10% buffer to account for sim-to-real errors.

Table X. STPA products for Finding 2.

#	UCA	Causes	Loss Scenario	Solution

LS-2.1	<Auto Trips> do not [terminate autonomy] when it exceeds the cleared flight envelope.	Control algorithm: Safety mechanisms coded with the wrong envelope values and cannot be changed.	UAV exceeds the cleared altitude/airspeed envelope but remains within the advertised envelope, and the UAV safety mechanisms do not trigger because they enforce the full advertised envelope. However, due to errors in predicting the full envelope, UAV is unstable outside of the cleared envelope and departs controlled flight or experiences structural damage. [L-1 – L-6]	Develop new mechanism to enforce cleared envelope.

Table XI. STPA products for Finding 3.

#	UCA	Causes	Loss Scenario	Solution
LS-3.1	<Human pilot> does not [fly] the <host UAV> when they are in control	Cause 1 (Process model): Human pilot believes the autonomy is in control, because the autonomy mode switch on the ground terminal is on.	Autonomy is not actually in control and the autonomy mode logic is in the process of timing out, but the human pilot can't see this, they only see that autonomy mode is enabled. The human pilot does not maneuver the UAV to avoid an obstacle and a collision occurs [L-1 – L-6].	Make autonomy mode timeout very short (~less than 3 seconds). Requires good timing between the human pilot and the autonomy operator that can be orchestrated using voice comms in the control room. Show the autonomy mode timeout timer on the UAV control station so the human pilot knows when the autonomy is not actually sending commands.
LS-3.2	<UAV FC> does not [disable autonomy mode] when autonomy is not sending commands	Cause 1 (Feedback): UAV FC is still receiving request for control but is not receiving any other messages.	The UAV receives no control inputs from the autonomy, so no one is flying the UAV until the autonomy is terminated by the human pilot or an auto safety trip. [L-3]	Add a check within the autonomy that makes the autonomy terminate itself if it is not sending commands to the UAV. Modify autonomy mode logic to check for autonomy commands, not just request for control.

Table XII. STPA products for Finding 4.

#	UCA	Causes	Loss Scenario	Solutions
LS-4.1	<Transition Maneuver> does not [transition control] to	Cause 1: Environmental conditions (e.g. turbulence) or UAV condition (e.g. mechanical	Transition maneuver refuses to return control to the human pilot	Include a mechanism to override the transition

	<human pilot> when autonomy is terminated	failure) prevent the maneuver's target conditions from being reached. Cause 2 (Feedback): UAV Flight Computer does not send feedback to the transition maneuver controller that its conditions have been met (due to sensor fault, etc.)	because it is unable to stabilize the aircraft and satisfy the exit criteria. The human pilot is then unable to maneuver to avoid obstacles and other hazards. No further testing can be conducted and the UAV may collide with an obstacle or be emergency terminated. [L-1 – L-6]	maneuver before its conditions are met.
LS-4.2	<Transition Maneuver> provides [transition control to <human pilot> too early before the transition exit criteria are met (e.g., wings level)	Cause 1: Environmental conditions (e.g. turbulence) or UAV condition (e.g. mechanical failure) prevent the maneuver's target conditions from being reached. Cause 2: The timeout limit for completing the transition maneuver is reached	The transition maneuver attempts to stabilize the aircraft and satisfy the exit criteria, but the transition takes longer than normal due to a disturbance, mechanical failure, or other condition. The built-in timeout limit is reached, forcing a transition when the aircraft is not stable. The human pilot may not expect the transition to complete at that time since the transition maneuver was still in progress and the aircraft had not yet reached the expected state (e.g., wings level)	Include a mechanism to notify the human pilot of any forced transition in advance of the timeout being reached. Ensure the transition maneuver is tested in worst-case disturbances that challenge the transition completion time.

Table XIII. Other STPA products.

#	UCA	Causes	Loss Scenario	Solution
LS-5.1	<Autonomy> continues to [request control] after autonomy mode is disabled by the human pilot.	Control algorithm: Autonomy not coded to stop requesting control when disabled.	Autonomy is terminated by remote pilot or auto trip but continues to request Control because it was not internally disabled. Later, test team re-enables autonomy mode, the autonomy immediately starts controlling the UAV and begins flying the previous test point because it was never turned off.	• Ensure autonomy operator always terminates agent if autonomy is terminated. • Update autonomy code to automatically stop requesting UAV control when the UAV autonomy mode is disabled.