

System-Theoretic Analysis of Unsafe Collaborative Control in Teaming Systems

Andrew N. Kopeikin¹ and Nancy G. Leveson²
Massachusetts Institute of Technology, Cambridge, MA, 02139, USA

Natasha A. Neogi³
National Aeronautics and Space Administration, Hampton, VA, 23666, USA

The interactions that occur in human-teaming are inspiring novel aerospace designs aimed at improving how humans and machines, or multiple machines, work together. Unfortunately, current Systems Engineering processes are ill-equipped to handle these complex relationships and are unable to design and assure the safety for these systems. To close part of this gap, this paper introduces a novel system-theoretic analytical process to identify unsafe collaborative control actions. It is part of a broader set of techniques that extend the state-of-the-art in hazard analysis, System Theoretic Process Analysis (STPA), to systematically address collaboration. The method rigorously expresses the different ways multiple commands may be unsafe together. Using Systems Theory, it employs abstraction to manage the combinatorial complexity in enumerating control contributions from multiple collaborating components. An algorithm integrates these concepts into an end-to-end process and is supported by automation to enumerate, refine, prune, and prioritize unsafe combinations of control actions. The output of the method feeds the specification of system requirements to implement safety-guided design starting early in concept development. The process is demonstrated on a manned-unmanned aircraft teaming case study and finds new causal factors that were not previously found in a past hazard analysis of the same system.

Nomenclature

c	= controller
d	= number of unsafe combinations of control actions
E	= end of a control action
M	= set of all control actions provided by the team
N	= set of all controllers in a team
p	= number of control actions to shared process
S	= start of a control action
u	= control action

I. Introduction

Human teams exhibit complex interactions that involve establishing roles, adapting functional responsibilities, maintaining shared cognition, coordinating, and helping one another close control loops [1]. These mutually influential relationships allow teammates to leverage each other's contributions to joint activity to improve performance in a task or to achieve something they cannot do alone. By contrast, the interactions observed between humans and automation in fielded aerospace applications are far simpler. However, there is significant interest in being able to incorporate

¹ PhD Graduate, Aeronautics and Astronautics, MIT, Technical Staff, MIT Lincoln Lab, and AIAA Member.

² J.C. Hunsaker Professor of Aeronautics and Astronautics, MIT.

³ Acting Senior Technologist for Intelligent Flight Systems, Subproject Manager for NASA System-Wide Safety Project, NASA Langley Research Center, and AIAA Associate Fellow.

the more complex team-inspired interactions into these systems to improve how human and machines, or multiple machines, work together [2].

Many teaming related aerospace system concepts have been proposed in the past decade. The civil aviation industry is exploring how to partner humans with future automated controllers that would be certified to safely take on some of the operating responsibilities for Urban Air Mobility (UAM) aircraft [3] or augment reduced crew operations in airliners [4]. Aviation authorities are researching how distributed decision-making and dynamic switching between humans and automated controllers can help integrate Unmanned Aircraft Systems (UAS) traffic into the National Airspace System [5], [6].

Other aerospace sectors are attracted by similar concepts. The space industry is turning to human-robot partnerships to improve deep space and planetary exploration [7]. The military is investigating how to use collaborative unmanned systems to deliver more overwhelming effects [8]. Several operational concepts team human-piloted aircraft with UAS to jointly execute complex missions [9], [10]. Finally, numerous multi-UAS prototypes have demonstrated how collaborative machines can achieve collective effects [11], [12].

Despite all of the interest in aerospace systems that implement team-like interactions, few have actually been fielded. One of the key challenges is that existing Systems Engineering processes do not adequately address how to design and assure the safety of these systems [13]–[16]. To do so, requires both (1) a theoretical framework to describe collaborative interactions that is consistent with the principles of modern safety-engineering and (2) a process built on that foundation to do analysis.

The overall research objective is to develop a rigorous and systematic approach to analyze the safety of collaborative control systems. Recently, this effort introduced a framework grounded in Systems Theory that defines a taxonomy of the structure of controller interactions and a set of dynamics observed in collaborative control [2]. The current work builds off of this framework to extend the state-of-the-art hazard analysis technique, System-Theoretic Process Analysis (STPA). Several analytical extensions are in development, which collectively are referred to as STPA-Teaming (Fig. 1).

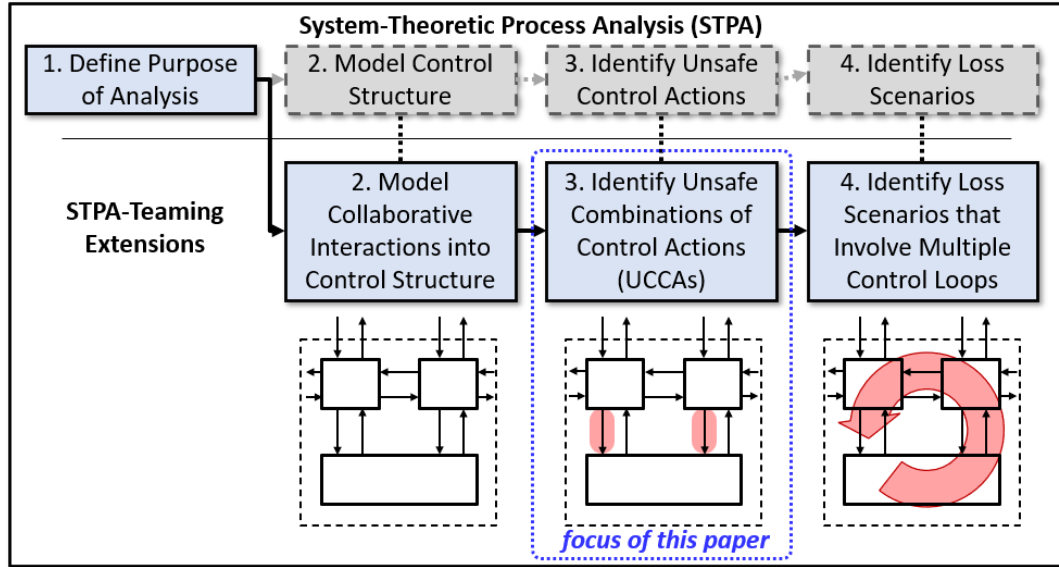


Fig. 1 Three Analytical Extensions Involved in STPA-Teaming

This paper specifically introduces a novel method to comprehensively determine how combinations of control actions from controllers collaborating as a system may be unsafe. This is a key capability needed to identify causal factors that contribute to those unsafe control outputs, and then derive system requirements to eliminate or mitigate them in design. The theoretical foundation of the technique is described, and then the approach is demonstrated on a Manned-Unmanned Teaming (MUM-T) case study involving a human piloted military aircraft partnering with unmanned aircraft to execute a mission.

The remainder of this paper is organized as follows. Section II reviews some of the limitations in common safety analysis techniques and it provides an argument for the selected approach. Section III introduces the collaborative system in the case study to then illustrate the new analytical approach. Section IV develops the novel approach to

identify unsafe control in collaborative systems. Section V applies the new technique to the case study and describes some of the results obtained. Future work in development for this research is previewed in Section VI.

II. Background

The community is becoming increasingly aware of some of the difficulties associated with expanding the nature of component interactions in aerospace systems. Several key challenges are recurrent in recent studies that have explored how to integrate teaming into the aviation domain [13]–[18].

First, the attributes needed to facilitate safe teaming are challenging. System components (humans and machines) may need to handle context-dependent dynamics such as establishing roles, maintaining shared situational awareness, mutual monitoring, and mutual assistance in a joint control problem. These dynamics are further complicated by inherent differences in how controllers interpret information and time, especially when humans partner with machines.

Another key challenge is that novel systems, especially those with complex collaborative interactions are difficult to address with current assurance processes. Typically, these processes are applied too late in the Systems Engineering lifecycle to effectively influence the design, and they are inadequate to handle interactions between software, hardware and humans [19]. The following reviews existing hazard analysis techniques, which underpin assurance, and explains why a system-theoretic approach was selected for this research.

A. Common Hazard Analysis Techniques

Hazard analysis is a process to systematically identify causal factors that can lead a system to enter a hazardous state. Its output is the foundation for other assurance processes, including definition of system requirements and constraints to enforce safety, implementation of these requirements, V&V, and documenting results for certification. For this reason, hazard analysis should begin influencing system design early in its conceptual stages, on the left side of the Systems Engineering “V”. However, in practice, this does not happen until much later [19].

The different hazard analysis techniques are founded on causality models, which impose patterns on observed events and represent assumptions about how the world operates [20]. In safety engineering, these models help explain why accidents occur. The majority of hazard analysis techniques used in traditional safety assurance programs are based on *Linear Chain of Events* causality models. Linear causality implies that if variable A has a causal influence on B , then B has no influence on A . While this assumption makes it easier to identify the “chains of failures” often reported in accident reports, it is an oversimplification that actually hinders safety engineering [19]. The most common techniques built on this model include Functional Hazard Analysis (FHA), Failure Modes and Effects Analysis (FMEA), Failure Modes and Effects Criticality Analysis (FMECA), and Fault Tree Analysis (FTA) [21]–[24].

There are major limitations associated with these hazard analysis techniques. They all focus on component failures and do not consider how the interactions between fully functional components can lead to hazardous states [19]. The assumption that causality is linear is a significant weakness recognized by the community since cyclic relations are common in real systems [25], [26]. All of these methods are hardware-oriented, and none can effectively handle the contributions of human and software control to accidents [19].

B. System Theoretic Techniques

Unlike in chain of event techniques, *System Theoretic* models acknowledge causal loops exist between components, where variable A can influence B , and B can also influence A [27]. System-Theoretic Accident Model and Processes (STAMP) is one such causality model, which treats safety as a control problem rather than a failure problem [19]. STAMP is the foundation for hazard analysis techniques such as System-Theoretic Process Analysis (STPA) and the STPA-Teaming extensions developed in this work.

STPA is a qualitative hazard analysis technique built upon the STAMP causality model [28]. Its foundation in Systems Theory overcomes many of the limitations of other methods. For this reason, this technique is becoming increasingly popular in many industries, including aerospace. It is executed in the four steps listed at the top of Fig. 1. The following is a simple illustrative example of STPA applied to a human operator supervising a multi-UAS system [29].

In Step 1 the system *losses* that are unacceptable to the stakeholders are identified. For the multi-UAS system, these may include (L1) loss of mission, (L2) loss of life or permanent disabling injury, and (L3) loss or damage to the UAS or equipment. The system-level *hazards*⁴ are defined and traced to the losses. Example hazards are (H1) UAS does not complete mission objectives [L1, L2], and (H2) structural integrity of UAS is violated [L3] [29].

⁴ A hazard is “a system state or set of conditions that, together with a particular set of worst-case environmental conditions, will lead to a loss” [28].

Step 2 models the system as a STAMP hierarchical control structure, in which responsibilities, control actions, and feedback elements are defined for each controller. A simplified adaption of the multi-UAS system control structure is in Fig. 2. It includes a human operator controlling an automated system that manages the fleet of UAS. For simplicity, the model is only labeled with one of its many control actions, the “Approve Course of Action (COA)” command provided by the operator to the fleet controller. The feedback relevant to that command is also shown.

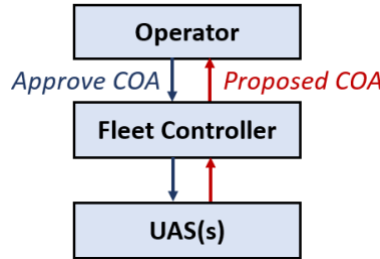


Fig. 2 STPA Example Multi-UAS System Control Structure (adapted from [29])

Step 3 identifies unsafe control actions (UCAs), which are control actions that, in a particular context, and worse-case environment, will lead to a hazardous state [28]. UCAs are defined using a specific structure, which is further discussed and leveraged for the analytical extension in Section IV. One example UCA is:

UCA-1: Operator *provides too late* the action “Approve COA”, when the COA no longer fulfills mission objectives [H1, H2, H3] [29]

Step 4 identifies causal scenarios that can lead to the UCA being issued. This is accomplished by systematically exploring potential breakdowns in the feedback control loop in the model. The following example scenario is derived from UCA-1 above. From it, design constraints can be derived to eliminate or mitigate the UCA and causal scenario.

Scenario-1: The Fleet Controller updates the COA request so frequently that the operator cannot assess its validity before it is replanned. Thus, the operator is in a cycle of perpetual COA review. [UCA-1] [29]

C. Hazard Analysis of Collaborative Control Systems

There is an overwhelming recognition that new aerospace systems with interactions inspired by teams face significant challenges in V&V and certification. Despite this, the literature is relatively silent regarding the limitations associated with hazard analysis techniques for these systems. This is surprising given the critical role of hazard analysis in the other safety assurance activities.

NASA has demonstrated how to employ methods like FHA and STPA in new aviation applications like UAM and is still exploring different methods to best address these novel technologies [30], [31]. Belcastro et al. describe a need to evolve hazard analysis techniques to address distributed multi-UAS operations. However, their analysis is limited to the study of historical data to identify potential hazards for the analysis of future systems [32]. Baig et al. survey applications of FTAs and highlight how they fall short in human-machine interactions, leading them to use Fuzzy Logic or other augmentation mechanisms to account for the non-deterministic nature of humans [33].

There are several reasons why STAMP was chosen as the foundation for this work. By focusing on control rather than just component failures, like the other techniques do, STAMP can help explore how collaborative control may be unsafe. The non-linear causality it can handle is inherent in collaborative relationships. It provides a unified framework to explore the interactions between hardware, software, and human controllers, which is key for human-machine and multi-machine teams. It also allows the system to be modeled in its larger socio-technical context. Finally, its system-theoretic foundation allows abstraction to be used to manage complexity.

A growing body of work has applied STAMP-based techniques to aerospace systems with complex collaborative interactions [34]–[39]. However, these studies shy away from analyzing distributed control when multiple controllers can make individual control decisions over shared a controlled process. In addition, the conventional process of UCAs in STPA tends to focus the analysis on one controller and one controlled process at a time and less on the collections of control actions from multiple collaborating controllers. Finally, the lack of direction on how to handle collaboration creates both ambiguity in how to analyze such interactions and a risk of missing associated causal factors.

To begin to address these challenges, our most recent work identified nine different interactions that are observed in collaborative systems (see Fig. 3). Their definitions are grounded in STAMP and follow the principles of Systems Theory [2]. Techniques such as the one introduced in Section IV are now in development to address these dynamics.

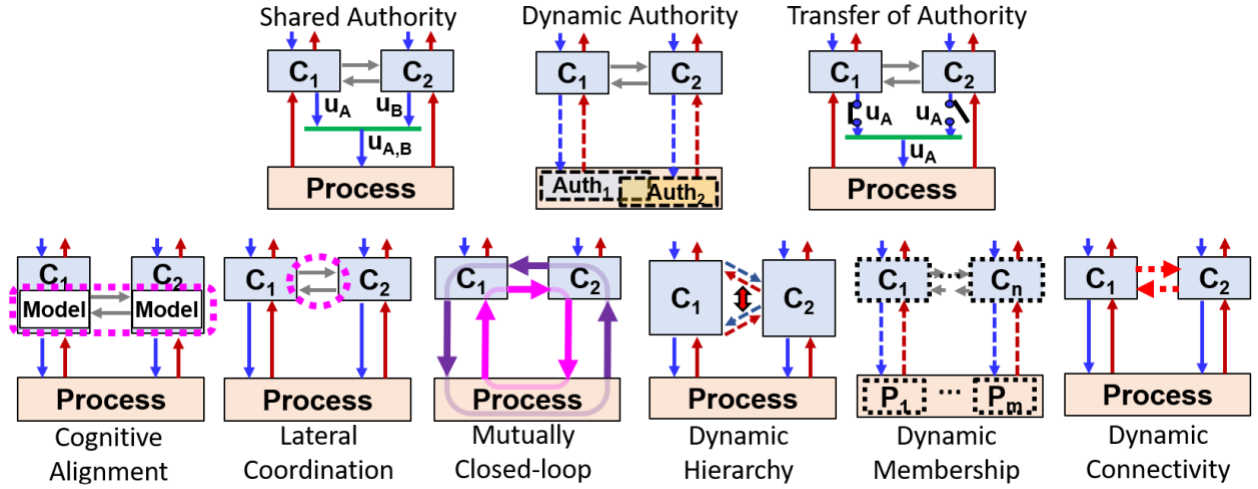


Fig. 3 System Theoretic Definition of Collaborative Control Interactions (adapted from [2])

III. Collaborative System Case Study: Manned-Unmanned Aircraft Teaming

The military is actively pursuing the development of multiple aviation concepts that aim to partner human piloted aircraft with unmanned aircraft [9], [10]. These novel systems involve the human pilot as the leader of the flight with one or more UAS acting as wingmen. Together they fly in formation and collaboratively execute complex mission tasks at the direction of the human.

The Manned-Unmanned Teaming (MUM-T) case study analyzed in this work was previously modeled and analyzed using STPA in several different studies [37]–[39]. The system described below is derived from Robertson’s detailed and comprehensive analysis work [37], which is henceforth referred to as *the baseline*.

The control structure for the MUM-T system is remodeled as shown in Fig. 4 to emphasize the shared authority of the human Team Lead (TL) and the UAS over the shared mission processes that include targets. Feedback and coordination information is hidden in the figure for simplicity. Each UAS can provide the *fix on target* (i.e., target designation in this work), *fire weapon*, and *search for target* commands. The TL tasks the UAS and can *fire* or *search* her/himself.

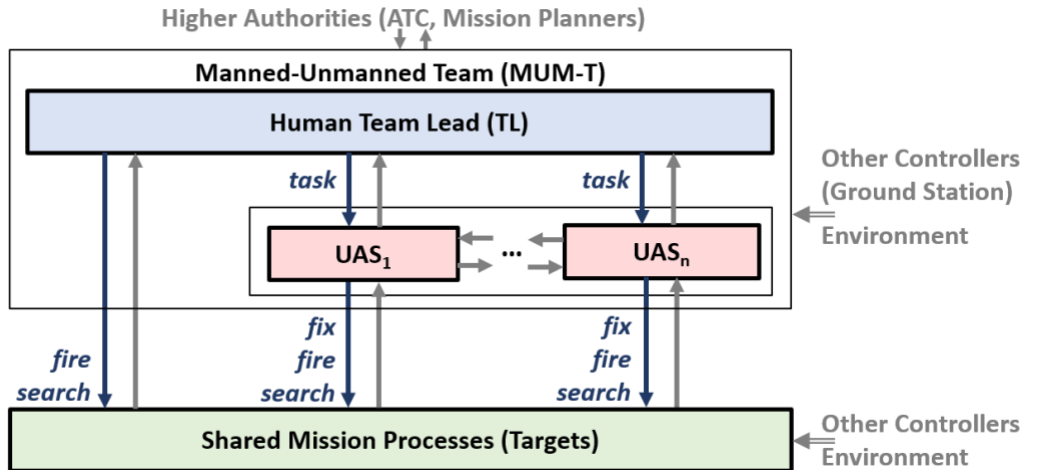


Fig. 4 Remodeled Control Structure of Manned-Unmanned Teaming System

The baseline assumes that it is hazardous for multiple controllers to simultaneously fix or fire on the same target. Similarly, to safely provide the *fire* command the target must be fixed. Multiple controllers may collaborate to engage a target, where one provides the *fix* and another *fires*.

Given this system model, a key question asked in STPA is “how can the controllers in the system issue control actions that may lead to a hazardous state?”. The method developed in the next section specifically determines what combinations of control actions by the overall MUM-T system are unsafe together.

IV. Identifying Unsafe Combinations of Control Actions (UCCAs)

Collaborative control fundamentally involves multiple controllers working together to control a shared process. A key implication of this *shared authority*, as shown in Fig. 3, is that the control actions from collaborating controllers cannot be analyzed individually. There may be unsafe causal factors that can only be identified when the actions of these multiple controllers are analyzed together.

Collaborative control may also involve *dynamic authority* (Fig. 3) that enables multiple controllers to adjust task allocation during execution. However, that may result in causal factors associated with control gaps, conflicts due to overlaps, or controller-task mismatches. Similarly, the system may allow for *transfer of authority* (Fig. 3) that can lead to unsafe control handoffs.

STPA currently lacks a way to comprehensively consider how multiple control actions may be unsafe together. The following subsections introduce a process that addresses this gap by identifying Unsafe Combinations of Control Actions (UCCAs), and which has the following attributes. First, it follows a systematic approach derived from STPA to ensure all relevant types of combinations are considered. Second, it leverages multiple levels of abstraction to manage combinatorial complexity. Finally, the UCCAs identified provide a pathway to then explore unsafe causal factors framed by the collaborative control dynamics defined in Fig. 3.

A. Types of UCCAs

One of the strengths of STPA is its systematic process to consider how a control action, given a particular context, can lead to a hazardous state. UCAs are identified using the specific structure shown in Fig. 5 [28].

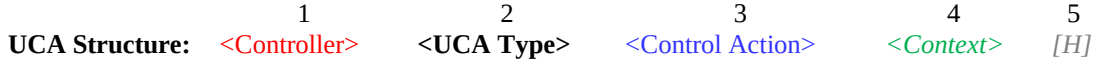


Fig. 5 Structure of a UCA in STPA

The controller (item 1) and the control action (item 3) are obtained from the control structure. STPA defines four types of UCAs (item 2), listed below, in which a control action may be unsafe. The set of four types is provably complete to describe a given control effort [28].

- UCA Type 1: not providing the control action
- UCA Type 2: providing the control action
- UCA Type 3: providing the control action too early or too late
- UCA Type 4: providing the control action for too long or stopping it too soon

The structure in Fig. 5 allows items 1-3 to be machine enumerated. This allows a human analyst to focus on determining if there is a context for the UCA (item 4), and if so, trace it to the hazard(s) it leads to (item 5).

The process to identify Unsafe Combinations of Control Actions (UCCAs) builds on the approach for UCAs. The UCA structure is expanded to incorporate combinations of control actions. Different types of UCCAs are derived from the types of UCAs to maintain the rigor provided by STPA. The resulting formulation provides the foundation necessary to algorithmically enumerate these combinations. As such, the potential UCCAs can be machine-generated, and again, focus the human analyst on specifying the context in which they are unsafe.

Two UCCA types are defined. The first, UCCA Type 1-2, combines UCA Types 1 and 2, which describe whether or not a control action is provided at all. Type 1-2 UCCAs help find contexts in which providing none, some, or all of multiple control actions is unsafe. As a simple example, consider two controllers c_i and c_j , that can each provide control action u_a and u_b respectively. Potential Type 1-2 UCCAs can be enumerated as follows, using an extended UCA structure with the same color coding as in Fig. 5:

Type 1-2 UCCA Example:

1. c_i does not provide u_a and c_j does not provide u_b when... [H]
2. c_i does not provide u_a and c_j provides u_b when... [H]
3. c_i provides u_a and c_j does not provide u_b when... [H]
4. c_i provides u_a and c_j provides u_b when... [H]

Second, the Type 3-4 UCCA combines UCA Types 3 and 4, which assume that the control action is provided, and instead focuses on the temporal sequence in which it occurs. Specifically, a Type 3 UCA considers when the control effort starts, rising from OFF to ON. A Type 4 UCA accounts for when the control effort ends, falling from ON to OFF. For a discrete command, not provided over time, only the rising edge is considered and Type 4 UCAs do not apply [28].

The resulting Type 3-4 UCCAs consider the temporal sequence of multiple control actions in relation to one another. They explore how starting or ending a control action before or after starting or ending another control action may be unsafe. The Type 3-4 UCCAs are enumerated using the same extended UCA structure described in the above example.

Type 3-4 UCCA Example:

1. c_i starts providing u_a before c_j starts providing u_b when... [H]
2. c_i starts providing u_a before c_j ends providing u_b when... [H]
3. c_i ends providing u_a before c_j starts providing u_b when... [H]
4. c_i ends providing u_a before c_j ends providing u_b when... [H]
- ...
8. c_j ends providing u_b before c_i ends providing u_a when... [H]

The two types of UCCAs provide the foundation to machine enumerate all possible control combinations. However, that does not by itself solve the problem of analyzing how multiple control actions together are unsafe. Additional processes are needed to handle combinations that are more complex than the two control actions considered above.

The MUM-T system (Fig. 4) exemplifies the combinatorial complexity that occurs when enumerating UCCAs. In that example, the Team Lead (TL) can provide 2 different actions, and each UAS can provide 3 actions, so there are $p = 8$ total control actions that can be provided to the shared process. More generally, p is found using Eq. (1), where $U_a(c_i) = 1$ if controller c_i can provide control action u_a , and $\mathbf{N}$ and $\mathbf{M}$ are the domains of all control actions and controllers respectively.

$$p = \sum U_i(c_a); \forall i \in \mathbf{N}, \forall a \in \mathbf{M} \quad (1)$$

There are $d^{1,2} = 2^p$ different possible Type 1-2 UCCA combinations of n controllers *providing or not providing* each of the m control actions. In the MUM-T example, this means there are $d^{1,2} = 2^8 = 256$ Type 1-2 UCCAs to enumerate. This number grows exponentially with n and m . The number of Type 3-4 UCCA permutations, or ordered sequences, in which up to any of these p control actions can be *started and ended* relative to one another grows even faster, as found in Eq. (2). For the MUM-T system, there $d^{3,4} = 17 \times 10^6$ possible Type 3-4 UCCAs.

$$d^{3,4} = \sum_{k=2}^p \frac{2^k p!}{(p-k)!} \quad (2)$$

As shown, teams of even modest sizes produce too many potential UCCAs if fully enumerated. It is not practical for a human analyst to identify the context(s) in which all these UCCAs are unsafe, and then develop causal scenarios to explain how they could occur. Simplification is necessary.

B. Using Abstraction to Manage Complexity

The ability to abstract models to manage complexity is one of the key strengths of system-theoretic approaches. As shown below, collaborative systems can be systematically abstracted to reduce the combinatorial complexity associated with UCCAs. The approach, summarized for a general collaborative system in Fig. 6, helps identify the context in which multiple control actions are unsafe together at a higher level of abstraction, where fewer combinations can be enumerated. Then, only those combinations found to be unsafe need to be further explored through refinement.

The approach begins with two different model representations, Abstractions 1a and 1b, which are shown one level up from the general collaborative system in Fig. 6. Each one focuses on a different way to combine multiple control actions. Abstraction 1a combines the multiple controllers that share authority over a process into one collective controller. This helps identify the unsafe combinations of different types of control actions issued by the overall team to the process. Conversely, Abstraction 1b maintains the multiple controllers, but it focuses only on one common type of control action at a time. It helps determine the unsafe combinations of controllers issuing a common control action.

At the level of Abstraction 1a and 1b the number of UCCAs grows slower than in the fully refined problem. But it still scales exponentially in the number of different types of control actions m for 1a and in the number of controllers n for 1b. To linearize the growth, each model is further abstracted to the two models at the top of Fig. 6, Abstractions 2a and 2b.

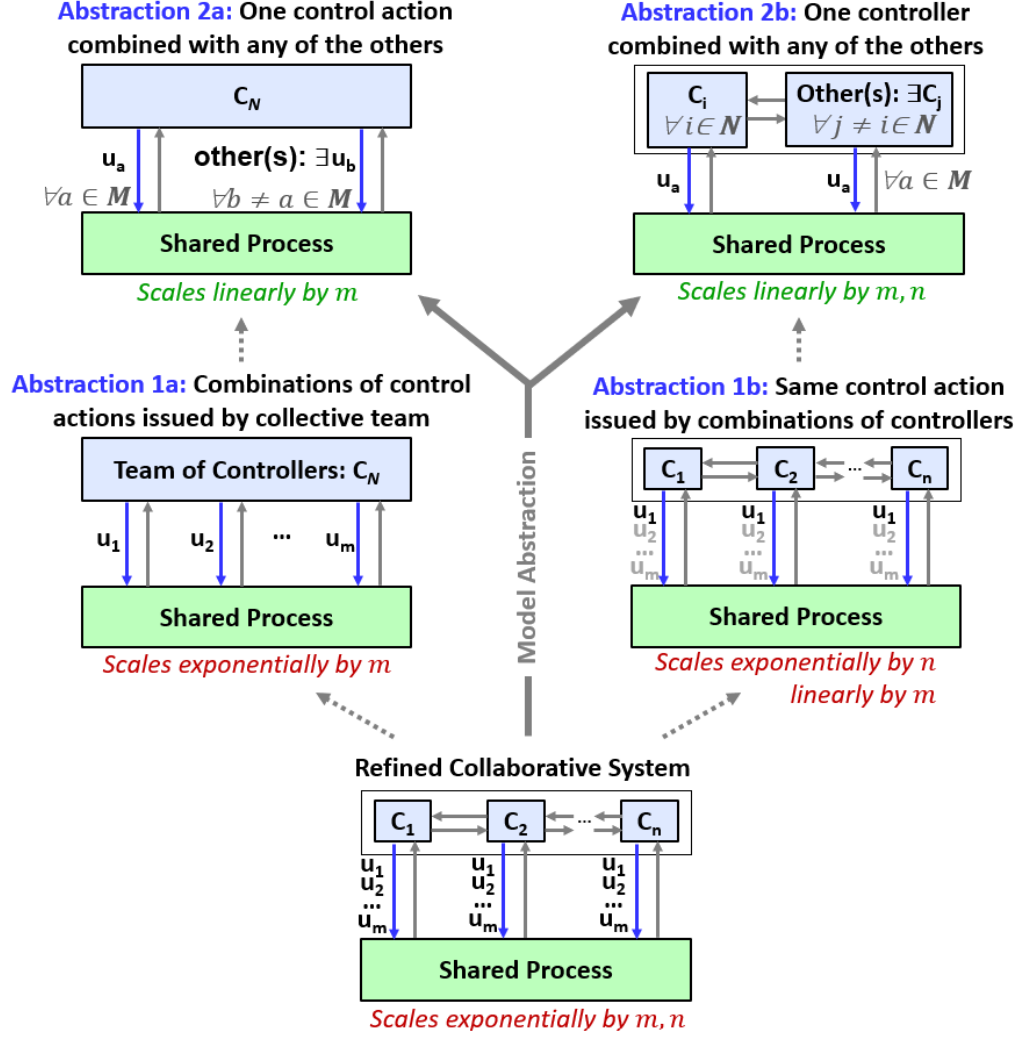


Fig. 6 Managing the Combinatorial Complexity of Identifying UCCAs Using Abstraction

Abstraction 2a is based on Abstraction 1a, and it iteratively considers any one of the control actions in combination with any of the others abstracted together. At this level, Type 1-2 UCCAs explore how the collective team provides (or does not provide) any one control action and provides (or does not provide) any of the others. One iteration of this process for a general three control action system, like the MUM-T example, for control action u_1 , is:

1. c_N **does not provide** u_1 and **does not provide** $\{u_2 \text{ and/or } u_3\}$ when... [H]
2. c_N **does not provide** u_1 and **provides** $\{u_2 \text{ and/or } u_3\}$ when... [H]
3. c_N **provides** u_1 and **does not provide** $\{u_2 \text{ and/or } u_3\}$ when... [H]
4. c_N **provides** u_1 and **provides** $\{u_2 \text{ and/or } u_3\}$ when... [H]

The same set of potential Type 1-2 UCCAs for Abstraction 2a of the MUM-T system is shown in Table 1 using a format inspired by truth tables found in propositional logic, where $\neg u$ means “not provide u command”. Two additional iterations, similar to those above, are needed compare u_2 with $\{u_1 \vee u_3\}$ and u_3 with $\{u_1 \vee u_2\}$.

Table 1 Abstraction 2a Type 1-2 UCCAs for MUM-T Example (Partial Set)

#	c_N		#	MUM-T Team		Context
1	$\neg u_1$	$\neg\{u_2 \vee u_3\}$	1	$\neg\text{fix}$	$\neg\text{any in}\{\text{fire, search}\}$	when...
2	$\neg u_1$	$\{u_2 \vee u_3\}$	2	$\neg\text{fix}$	any in{fire, search}	when...
3	u_1	$\neg\{u_2 \vee u_3\}$	3	fix	$\neg\text{any in}\{\text{fire, search}\}$	when...
4	u_1	$\{u_2 \vee u_3\}$	4	fix	any in{fire, search}	when...

Type 3-4 UCCAs follow a similar concept. They represent ways in which the collective team may start or end any one control action before starting or ending any of the others. One of Type 3-4 UCCAs in Abstraction 2a is listed in table format for the MUM-T example is provided in Table 2. In the table, $S(u)$ means “start control action u ”, $E(u)$ means “end u ”, and F is the Linear Temporal Logic operator for *Some Future Step*. Additional iterations are required for u_2 and u_3 , and the reverse sequences such as $S(\{u_2 \vee u_3\}) F S(u_1)$ can also be considered.

Table 2 Abstraction 2a Type 3-4 UCCAs for MUM-T Example (Partial Set)

#	c_N	$F c_N$	#	Team	before Team	Context
1	$S(u_1)$	$S(\{u_2 \vee u_3\})$	1	starts fix	starts any in{fire, search}	when...
2	$S(u_1)$	$E(\{u_2 \vee u_3\})$	2	starts fix	ends any in{fire, search}	when...
3	$E(u_1)$	$S(\{u_2 \vee u_3\})$	3	ends fix	starts any in{fire, search}	when...
4	$E(u_1)$	$E(\{u_2 \vee u_3\})$	4	ends fix	ends any in{fire, search}	when...

These tables can be generated using automation. The human analyst can then evaluate each item and determine if there is one or more context(s) in which that combination can lead to a hazard. They must also specify what control actions in the abstracted set are relevant to each UCCA context. For example, item 3 in Table 1 produces the following UCCA, where the search command is not relevant to the context:

UCCA 3: MUM-T does not fix and provides fire when target fired-on must be fixed [H3, H5].

Abstraction 2b follows a similar logic as in 2a and uses Abstraction 1b as a foundation to consider how any one controller in combination with any of the others abstracted together may issue a common control action (see Fig. 6). Here, Type 1-2 UCCAs explore ways the one controller and the others provide or do not provide the action, as shown in Table 3. The process is reiterated for each control action.

A key difference in how UCCAs are defined in Abstraction 2b, in this work, is that the specific controllers do not need to be listed. For the systems explored in this research, it is assumed that the context in which a control action, or combination of control actions, is unsafe is agnostic to which specific controller provides it. As such, the controllers listed in the UCCA can be generalized, which reduces the number of combinations that need to be analyzed.

Table 3 Abstraction 2b Type 1-2 UCCAs for MUM-T Example

#	$U_1(x)$		Common Issues	#	fire provided by		Context
1	$\neg U_1(c_i)$	$\neg U_1(\{c_{j1} \vee c_{j2}\})$	gap	1	$\neg$ any one	$\neg$ any of others	when...
2	$U_1(c_i)$	$\neg U_1(\{c_{j1} \vee c_{j2}\})$	mismatch	2	any one	$\neg$ any of others	when...
3	$U_1(c_i)$	$U_1(\{c_{j1} \vee c_{j2}\})$	overlap	3	any one	any of others	when...

*Does not need to be considered, included in items 2 and 3 (see discussion)

However, it is important to note that the specific controllers are considered in the refinement of the UCCA. This step, later illustrated in Table 5, explores how those different controllers can contribute to that higher-level output. By listing these options, the causal factors that are controller specific can later be identified in the last step of the hazard analysis, scenario development. The refinement of a UCCA can be fully automated as described in the next subsection.

Type 3-4 UCCAs in this abstraction explore the sequences in which any one controller and any other start or end a common control action. Table 4 shows the process for one control action (u_1), and similar iterations are necessary for each of the other control actions.

Table 4 Abstraction 2b Type 3-4 UCCAs for MUM-T Example

#	$U_1(x)$	$F U_1(x)$	Common Issues	#	search by	before search by	Context
1	$S(U_1(c_i))$	$S(U_1(c_j))$		1	any one starts	any other starts	when...
2	$S(U_1(c_i))$	$E(U_1(c_j))$	handoff overlap	2	any one starts	any other ends	when...
3	$E(U_1(c_i))$	$S(U_1(c_j))$	handoff gap	3	any one ends	any other starts	when...
4	$E(U_1(c_i))$	$E(U_1(c_j))$		4	any one ends	any other ends	when...

Using Abstraction 2b, the systematic consideration for how different controllers provide, don't provide, start and end a particular command highlights potential gaps, overlaps, mismatches, and handoff issues that are common in shared control, as shown in Table 3 and Table 4. This is a key strength of the overall UCCA approach.

C. UCCA Identification Algorithm

The concepts developed in the previous subsections are now integrated into an end-to-end algorithm to identify Unsafe Combinations of Control Actions (UCCAs). Several new ideas are incorporated to reduce and prioritize the output set of UCCAs and improve the efficiency of scenario development. The UCCA Identification Algorithm, expressed below in pseudocode form, includes portions that are automated (denoted by *) and others performed by a human analyst (**). A prototype tool that executes the automated portions of Algorithm 1 was developed in MATLAB and is employed in the case study.

Algorithm 1: UCCA Identification (Pseudocode)

Input: Authorities (each controller & its control actions), Interchangeable controllers, Special Interactions

Output: sets of UCCAs (abstracted & refined)

1. *Enumerate* the possible abstracted UCCAs*
 2. *for each* Abstracted UCCA**
 3. *if there is an unsafe Context***
 4. *Insert* the Context, *List* the relevant control actions**
 5. *for each* Abstracted UCCA with an unsafe Context*
 6. *Refine* the UCCA*
 7. *Prune* equivalent refined UCCAs*
 8. *Prioritize* remaining refined UCCAs*
 9. **return** UCCAs (abstracted & refined)*
- // * Step automated; ** Step performed by human
-

The first input to the algorithm is the set of control actions each controller can provide to the shared process, as defined in the control structure. The second input is the set of controllers considered interchangeable, which is used to prune refined UCCAs that are considered equivalent to others in terms of potential causal factors, as discussed later. The third input is a set of special interactions that can be encoded to add or remove refined UCCAs when enumerated, as also addressed later.

Line 1 in the algorithm enumerates all the combinations of control actions using a function that implements the procedure introduced in the previous subsection. This process is fully automated. The MATLAB prototype developed to demonstrate this concept automatically produces four tables with the control combinations and placeholders for an analyst to enter potential contexts and other information described in the next step of the Algorithm.

In Lines 2-4, a human analyst determines for each potential UCCA produced by the automation if there is a context, or multiple contexts, in which that control combination is hazardous. They must also designate the control actions that are relevant to the context. As an example, in Table 5, search is not relevant. This part of the algorithm benefits from human expertise and intuition across multiple domains.

Table 5 Refinement, Pruning, and Prioritization of UCCAs

UCCA	Team	Team	Context		
3	¬fix	{fire ∨ search}	<i>when target fired-on must be fixed [H3, H5]</i>		

Refined	TL	UAS ₁	UAS _n	Pruned?	Priority
3.1	-	-	fire	No	High
3.2	-	fire	-	Yes (3.2 ≡ 3.1)	N/A - Pruned
3.3	fire	-	-	No	High
3.4	-	fire	fire	No	Low
3.5	fire	-	fire	No	Low
3.6	fire	fire	-	Yes (3.6 ≡ 3.5)	N/A - Pruned
3.7	fire	fire	fire	No	Low

In Line 6, the automation refines each abstracted UCCA with an unsafe context. It does so by finding every combination of specific controllers that can contribute to the collective control output listed in each abstracted UCCA with an unsafe context. The function looks for equivalence of the union of control actions provided by all controllers into that specified in the abstracted UCCA. In simple terms, if a control action is provided in the abstracted Type 1-2 UCCA, equivalence is achieved if any controller in the refined set provides that control action.

Special interactions may also be defined to influence how equivalence is determined. For instance, if multiple controllers provide a command u , they may interfere with each other and yield a collective output equivalent to $\neg u$ no command being provided. In such cases, if the UCCA includes the team providing $\neg u$ the refined set can involve cases with multiple controllers providing u .

In Line 7, the automation prunes refined UCCA combinations that are considered equivalent to others previously listed on the basis of interchangeable controllers. In this example, the UAS are considered interchangeable. Causal scenarios will be no different if it is UAS_1 that provides part of the UCCA, or instead, another UAS. As a result, items 3.1 and 3.2 in Table 5 are equivalent as they both involve one UAS providing the *fire* command. No additional information is gained from 3.2 if 3.1 is analyzed, and as such 3.2 is pruned from the set.

In Line 8, the automation optionally prioritizes the refined UCCAs to help focus the rest of the hazard analysis, as shown in Table 5. The goal is to highlight the UCCAs that will contribute more new information in causal scenario analysis and devalue those likely to provide mostly repetitive information. The prioritization scheme implemented in this work is a heuristic formed on engineering judgment, but it could be implemented differently based on other application needs.

The concept is illustrated using again the $\neg fix$ and *fire* UCCA refined in Table 5. Scenarios for this UCCA must explain why (1) any controller would provide *fire* when (2) no controller provides *fix*, and vice versa. It is less important in the context of the UCCA to understand why multiple controllers would provide *fire*. As such, in the remaining set of refined UCCAs that were not pruned, those with one controller providing *fire* are prioritized over those with multiple controllers issuing that command.

Finally, the set of UCCAs refined, pruned, and prioritized using the automation is returned by Algorithm 1. The UCCAs, both abstracted and refined, are now ready for the analyst to use in the development of causal scenarios.

V. Results and Discussion

The Unsafe Combination of Control Actions (UCCAs) Identification Algorithm was executed on the MUM-T case study with the assistance of a tool prototyped in MATLAB. The automation creates four tables that include all the enumerated abstracted UCCAs. These are then reviewed by the human analyst for unsafe contexts. Tables 6-9 show examples of the abstracted UCCAs that were specified using this process.

Table 6 MUM-T Abstraction 2a Type 1-2 UCCAs (Example Set)

id	team	team	context	rel*
1	$\neg fix$	$\neg\{fire, search\}$	when there are mission tasks to execute [H3]	2,3
2	fix	$\neg\{fire, search\}$	when target will compromise mission if fixed but not fired on [H3]	2
4	fix	$\{fire, search\}$	when fixed target is different than target fired-on [H3, H5]	2

*rel are the control actions relevant to the UCCA in column 3. 1 = fix, 2 = fire, 3 = search

Table 7 MUM-T Abstraction 2a Type 3-4 UCCAs (Example Set)

id	team ...	then ...	context	relevant
15	E(fix)	S(fire, search)	when target fired on must be fixed [H3, H5]	fix, fire
17	S(fire)	S(fix, search)	when target fired on must be fixed [H3, H5]	fix, fire

Table 8 MUM-T Abstraction 2b Type 1-2 UCCAs (Example Set)

id	C_i	$\exists C_j$	context
37	$\neg fix$	$\neg fix$	when there is a priority target to engage & a teammate can fire [H3]
41	fire	$\neg fire$	when tasked entity is not capable and another is [H3]
42	fire	fire	when that creates excessive effects [H3, H5]

Table 9 MUM-T Abstraction 2b Type 3-4 UCCAs (Example Set)

id	C_i ...	then C_j ...	context
47	S(fix)	E(fix)	when that creates mutual interference [H1, H3]
56	E(search)	S(search)	when that creates an excessive gap in the search [H3]

Based on 19 abstracted UCCAs found in analysis, the tool generated 63 refined, pruned, and prioritized UCCAs. An example of a Type 1-2 UCCA refined is provided in Table 5. UCCA 15 is refined in Table 10 to demonstrate the

process for a Type 3-4 UCCA. Item 15.2 is deprioritized here because only one controller, UAS₁, provides all the control actions involved in the unsafe combination, and as such does not involve collaborative control.

Table 10 Refinement of MUM-T UCCA 15 (Example for Type 3-4 UCCA)

id	TL	UAS1		UAS2	priority	context
15.1	F S(fire)	E(fix)			High	<i>when target fired on must be fixed [H3, H5]</i>
15.3		E(fix)		F S(fire)	High	
15.2		E(fix)	F S(fire)		Low	

As is the case with the UCAs produced in STPA, the UCCAs developed in STPA-Teaming can be directly formed into safety-constraints as part of system design requirements. For example, safety-constraints (SC) derived from UCCA 15 and its refined set may include:

- SC-1.1: The system must not end providing the fix on target command before it starts providing the fire weapon command. [UCCA-15]
 - SC-1.1.1: A UAS must not end a fix before a human piloted aircraft starts to fire [UCCA-15.1]
 - SC-1.1.2: A UAS must not end a fix before another UAS starts to fire [UCCA-15.2]
 - SC-1.1.3: A UAS must not end a fix before it starts to fire [UCCA-15.3]
- SC-1.2: If the system ends providing a fix before it starts to fire, it must detect this occurrence and take appropriate actions to ensure the target is not fired on without a fix. [UCCA-15]

These UCCAs also feed the development of loss scenarios that explain how the UCCA could occur. In this process, potential breakdowns in the involved feedback control loops are investigated to identify causal factors that could contribute to the UCCA. For example, one of many loss scenarios that could explain UCCAs 15.1 and 15.2 is:

Scenario-2: The <TL, UAS> tasked to fire does not receive feedback from the UAS providing the fix that the fire command should not be provided. For example, the controller tasked to fire has already fired one successful shot. The UAS providing the fix observes the target getting destroyed, stops providing the fix, and attempts to provide fire feedback to the controller firing. However, that feedback is not adequately received or interpreted by that controller (reasons for inadequate feedback can be refined). [UCCA-15.1,15.2]

Scenario-2 above is an example of a loss scenario that was not identified in the baseline analysis of MUM-T. It highlights how the new method specifically explores the collaborative control interactions between two controllers with *shared authority* over the mission process, which is one of the collaborative control dynamics identified in recent work [2]. The rigorous consideration of unsafe control combinations and the identification of such new scenarios can help specify new safety requirements so that the system is designed to be safe. These requirements are traceable to the unsafe control actions, to the hazards, and ultimately to the unacceptable losses they are intended to prevent.

VI. Conclusion

Systems Engineering processes are currently limited in their ability to design and assure the safety of aerospace systems that incorporate team-inspired interactions between humans and machines, or between multiple machines. This paper introduced one of a set of novel techniques needed to extend the state-of-the-art in hazard analysis to address collaborative control interactions. The method is grounded in STAMP, derived from STPA, and which systematically and rigorously analyzes unsafe combinations of control actions. It employs both abstraction and automation to enumerate, refine, prune, and prioritize the sets of collaborative commands that together can lead the system to enter a hazardous state.

This analytical process builds on previous work in which collaborative interactions were defined using Systems Theory [2]. The way in which the control combinations are enumerated using abstraction is predicated on the *shared authority* dynamic as explained in that work. The method also systematically finds common control problems that occur in *dynamic authority* and *transition of authority*, which include control gaps, conflicts, mismatches, and unsafe handoffs. The ability to find unsafe combinations of control actions aligned with these three dynamics is demonstrated in the Manned-Unmanned Teaming (MUM-T) case study. As a result, new causal factors can be found that were previously not systematically considered using the available STPA guidance.

Additional methods are currently in development to systematically consider causality in the remaining six collaborative control dynamics at the bottom of Fig. 3. One is a template for a generic collaborative control structure to be able to express these causal relationships explicitly in the system model. Another is a more systematic process to develop loss scenarios using the UCCAs found in this paper, which is framed by the collaborative control dynamics.

Finally, a framework will integrate safety-guided design and assurance processes using the outputs of STPA and STPA-Teaming, the collective of the new techniques. These will be presented in future work.

Acknowledgments

The authors are thankful for discussions with Dr. John Thomas, MIT, and his insights on STPA and how to define unsafe control. This research would not have been possible without the support of the MIT Lincoln Scholar Program.

References

- [1] M. Johnson, J. M. Bradshaw, P. J. Feltovich, C. M. Jonker, M. B. Van Riemsdijk, and M. Sierhuis, "Coactive Design: Designing Support for Interdependence in Joint Activity," *J. Hum.-Robot Interact.*, vol. 3, no. 1, p. 43, Mar. 2014, doi: 10.5898/JHRI.3.1.Johnson.
- [2] A. N. Kopeikin, N. G. Leveson, and N. A. Neogi, "Defining Collaborative Control Interactions Using Systems Theory," presented at the 33rd Annual INCOSE International Symposium, Honolulu, HI, Jul. 2023.
- [3] GAMA, "A Rational Construct for Simplified Vehicle Operations (SVO)." General Aviation Manufacturers Association EPIC SVO Subcommittee Whitepaper, May 20, 2019.
- [4] R. E. Bailey, L. J. Kramer, K. D. Kennedy, C. L. Stephens, and T. J. Etherington, "An assessment of reduced crew and single pilot operations in commercial transport aircraft operations," presented at the 2017 IEEE/AIAA 36th Digital Avionics Systems Conference (DASC), Sep. 2017, pp. 1–15. doi: 10.1109/DASC.2017.8101988.
- [5] "Next Generation Air Transportation System: Next Gen UAS Research, Development and Demonstration Roadmap." NextGen Joint Planning and Development Office, Mar. 15, 2012. [Online]. Available: <https://irp.fas.org/program/collect/uas-nextgen.pdf>
- [6] NASA, "NASA Aeronautics Strategic Implementation Plan." 2017. [Online]. Available: www.nasa.gov/sites/default/files/atoms/files/sip-2017-03-23-17-high.pdf
- [7] T. Fong, "Human-robot teaming for space exploration." NASA Intelligent Robotics Group, Oct. 11, 2017. [Online]. Available: ntrs.nasa.gov/api/citations/
- [8] B. Clark, D. Patt, and H. Schramm, "Mosaic Warfare: Exploiting Artificial Intelligence and Autonomous Systems to Implement Decision-Centric Operations." Center for Strategic and Budgetary Assessments (CSBA), Feb. 11, 2020. [Online]. Available: <https://csbaonline.org/research/publications/mosaic-warfare-exploiting-artificial-intelligence-and-autonomous-systems-to-implement-decision-centric-operations>
- [9] Lockheed Martin, "HAVE RAIDER Demo - U.S. Air Force, Lockheed Martin Demonstrate Manned/Unmanned Teaming," Feb. 06, 2018. www.lockheedmartin.com/en-us/capabilities/autonomous-unmanned-systems/unmanned-military-case-study-have-raider-demo.html (accessed Jan. 28, 2022).
- [10] V. Insinna, "Under Skyborg program, F-35 and F-15EX jets could control drone sidekicks," *Defense News*, May 22, 2019. www.defensenews.com/air/2019/05/22/under-skyborg-program-f-35-and-f-15ex-jets-could-control-drone-sidekicks/ (accessed Jan. 28, 2022).
- [11] DoD, "Perdix Fact Sheet." Department of Defense - Strategic Capabilities Office, 2017.
- [12] A. Hudson, "The Looming Swarm," *Air Force Magazine*, Mar. 22, 2019. www.airforcemag.com/article/the-looming-swarm/ (accessed Jan. 21, 2022).
- [13] L. J. Prinzel, "Human-Autonomy Teaming - A Review of Literature and Preliminary Recommendations." NASA Langley Research Center, Transformational Tools and Technologies, Autonomous Systems, 2019.
- [14] A. Pritchett, M. Portman, and T. Nolan, "Research & Technology Development for Human-Autonomy Teaming Final Report: Literature Review and Findings from Stakeholder Interviews," 2018.
- [15] M. M. Connors, "Concepts for the Design of Human-Autonomy Systems," *NASA Ames Res. Cent.*, 2017.
- [16] K. Kearns, "DoD Autonomy Roadmap - Autonomy Community of Interest," presented at the NDIA 19th Annual Science & Engineering Technology Conference, Mar. 2018. [Online]. Available: <https://ndiastorage.blob.core.usgovcloudapi.net/ndia/2018/science/Kearns.pdf>
- [17] J. Holbrook *et al.*, "Enabling Urban Air Mobility: Human-Autonomy Teaming Research Challenges and Recommendations," presented at the AIAA AVIATION 2020 FORUM, Jun. 2020. doi: 10.2514/6.2020-3250.
- [18] M. R. Endsley, "From Here to Autonomy: Lessons Learned From Human-Automation Research," *Hum. Factors*, vol. 59, no. 1, pp. 5–27, Feb. 2017, doi: 10.1177/0018720816681350.
- [19] N. G. Leveson, *Engineering a safer world: systems thinking applied to safety*. in Engineering systems. Cambridge, MA: MIT Press, 2011.
- [20] N. G. Leveson, *Introduction to System Safety Engineering*, In Production. Cambridge, MA: MIT Press, 2023.
- [21] SAE, "ARP-4761: Guidelines and Methods for Conducting the Safety Assessment Process on Civil Airborne Systems and Equipment." Society of Automotive Engineers, SAE ARP4761, SAE International, Warrendale, PA, 1996.
- [22] C. Ericson, *Hazard Analysis Techniques for System Safety*. John Wiley & Sons, Ltd, 2005.
- [23] FAA, "Advisory Circular AC 23.1309: System Safety Analysis and Assessment for Part 23 Airplanes." Department of Transportation - Federal Aviation Administration, 2011.
- [24] Department of Defense, "MIL-STD-882E - System Safety." May 11, 2012.
- [25] P. Spirtes, C. N. Glymour, R. Scheines, and D. Heckerman, *Causation, prediction, and search*. MIT Press, 2000.

- [26] D. M. Hausman, "Review Article: The Mathematical Theory of Causation," *Br. J. Philos. Sci.*, vol. 50, no. 1, pp. 151–162, 1999.
- [27] N. G. Leveson, *Safeware 2: Safety Engineering for a High-Tech World (Pre-published)*. 2021.
- [28] N. G. Leveson and J. P. Thomas, *STPA Handbook*. 2018. [Online]. Available: https://psas.scripts.mit.edu/home/get_file.php?name=STPA_handbook.pdf
- [29] E. B. Johnson, A. N. Kopeikin, N. G. Leveson, and A. W. Drysdale, "Hazard Analysis for Human Supervisory Control of Multiple Unmanned Aircraft Systems," presented at the 56th International Symposium on Aviation Psychology, 2021, p. 274.
- [30] K. Wasson, N. Neogi, M. Graydon, J. Maddalon, P. Miner, and G. F. McCormick, "Functional Hazard Assessment for the eVTOL Aircraft Supporting Urban Air Mobility (UAM) Applications: Exploratory Demonstrations," *NASA Rep.*, p. 68, 2020.
- [31] M. Graydon, N. A. Neogi, and K. Wasson, "Guidance for Designing Safety into Urban Air Mobility: Hazard Analysis Techniques," in *AIAA Scitech 2020 Forum*, Orlando, FL: American Institute of Aeronautics and Astronautics, Jan. 2020. doi: 10.2514/6.2020-2099.
- [32] C. Belcastro, R. Newman, J. Evans, D. Klyde, L. Barr, and E. Ancel, *Hazards Identification and Analysis for Unmanned Aircraft System Operations*. 2017. doi: 10.2514/6.2017-3269.
- [33] A. A. Baig, R. Ruzli, and A. B. Buang, "Reliability Analysis Using Fault Tree Analysis: A Review," *Int. J. Chem. Eng. Appl.*, pp. 169–173, 2013, doi: 10.7763/IJCEA.2013.V4.287.
- [34] A. Kharsansky, "A systemic approach toward scalable, reliable and safe satellite constellations," Master's Thesis, Massachusetts Institute of Technology, Cambridge, MA, 2020.
- [35] D. C. Horney, "Systems-Theoretic Process Analysis and Safety-Guided Design of Military Systems," Master's Thesis, Massachusetts Institute of Technology, Cambridge, MA, 2015.
- [36] K. E. Johnson, "Systems-Theoretic Safety Analyses Extended for Coordination," PhD Dissertation, Massachusetts Institute of Technology, Cambridge, MA, 2017.
- [37] J. Robertson, "System Theoretic Process Analysis Applied to Manned-Unmanned Teaming," Master's Thesis, Massachusetts Institute of Technology, Cambridge, MA, 2019.
- [38] K. L. Hobbs, C. Cargal, E. Feron, and R. S. Burns, "Early Safety Analysis of Manned-Unmanned Team System," in *2018 AIAA Information Systems-AIAA Infotech @ Aerospace*, Kissimmee, Florida: American Institute of Aeronautics and Astronautics, Jan. 2018. doi: 10.2514/6.2018-1984.
- [39] K. L. Hobbs *et al.*, "Systems Theoretic Process Analysis of a Run Time Assured Neural Network Control System." arXiv, Sep. 01, 2022. [Online]. Available: <http://arxiv.org/abs/2209.00552>