Hazard Analysis for Human-on-the-Loop Interactions in sUAS Systems

Michael Vierhauser
michael.vierhauser@jku.at
Johannes Kepler University Linz
Austria

Md Nafee Al Islam
Ankit Agrawal
Jane Cleland-Huang
JaneHuang@nd.edu
University of Notre Dame, USA

James Mason
james.mason@ngc.com
Northrop Grumman
USA

ABSTRACT

With the rise of new AI technologies, autonomous systems are moving towards a paradigm in which increasing levels of responsibility are shifted from the human to the system, creating a transition from human-in-the-loop systems to human-on-the-loop (HoTL) systems. This has a significant impact on the safety analysis of such systems, as new types of errors occurring at the boundaries of human-machine interactions need to be taken into consideration. Traditional safety analysis typically focuses on system-level hazards with little focus on user-related or user-induced hazards that can cause critical system failures. To address this issue, we construct domain-level safety analysis assets for sUAS (small unmanned aerial systems) applications and describe the process we followed to explicitly, and systematically identify Human Interaction Points (HiPs), Hazard Factors and Mitigations from system hazards. We evaluate our approach by first investigating the extent to which recent sUAS incidents are covered by our hazard trees, and second by performing a study with six domain experts using our hazard trees to identify and document hazards for sUAS usage scenarios. Our study showed that our hazard trees provided effective coverage for a wide variety of sUAS application scenarios and were useful for stimulating safety thinking and helping users to identify and potentially mitigate human-interaction hazards.

CCS CONCEPTS

• **Software and its engineering → Software safety**.

KEYWORDS

Human-sUAS interaction, safety analysis, hazard analysis, sUAS

ACM Reference Format:
Michael Vierhauser, Md Nafee Al Islam, Ankit Agrawal, Jane Cleland-Huang, and James Mason. 2021. Hazard Analysis for Human-on-the-Loop Interactions in sUAS Systems. In *Proceedings of the 29th ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering (ESEC/FSE '21), August 23–28, 2021, Athens, Greece*. ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/3468264.3468534

ESEC/FSE '21, August 23–28, 2021, Athens, Greece

ACM ISBN 978-1-4503-8562-6/21/08…$15.00
https://doi.org/10.1145/3468264.3468534

1 INTRODUCTION

Autonomous systems are increasingly moving towards a paradigm in which humans and machines work in tandem to achieve relatively complex tasks – typically in systems that are now referred to as "human-on-the-loop" (HotL) [50]. In contrast to a more traditional "human-in-the-loop" (HitL) system, in which a human makes decisions at key points of the system's execution, a HotL system exhibits far greater machine autonomy while providing situational awareness to humans. HotL environments are able to take full advantage of machine autonomy to perform targeted tasks efficiently and quickly; however, in addition to traditional hazards, they introduce the potential for new types of errors that occur at the boundaries of human-machine interactions. Current paradigms that explore these interactions in safety-critical systems fail to fully evaluate the way in which humans contribute to, impact, or fail to impact, system safety in small Unmanned Aerial Systems (sUAS).

Historically, many hazards have occurred at the human-CPS interface. For example, in 1988 the US Navy's USS Vincennes shot down a civilian plane with 290 people on board. The Vincennes had entered Iranian water and operators mistakenly identified the Airbus as an attacking F-14 Tomcat despite the fact that the Airbus was climbing and emitting appropriate civilian IFF signals. The mistaken identification was partially attributed to a user interface flaw which caused the operator to confuse the data of a military plane in the area with that of the civilian one [17]. Human operators are frequently blamed for these types of errors which have been widely reported as contributing factors in 60% to 85% of accidents in domains such as aviation and medical devices [49]. However, many of these "human" failures can be directly attributed to flaws in the underlying system design [42] and could therefore be classified as design-induced-faults [27].

Similar examples are emerging in the domain of small Unmanned Aerial Systems, such as the case of a near collision between an sUAS and a highway patrol helicopter in California in 2015. The sUAS was flying at over 700 feet even though, based on FAA regulations, the maximum altitude allowed was 400 feet. In this case, the RPIC (remote pilot in command) had deliberately set a higher RTL (return to launch) altitude to avoid electrical pylons. During flight the signal to the sUAS was lost, and the RTL failsafe mechanism activated causing the sUAS to return home at an illegal altitude, resulting in a near collision with the helicopter. While the RPIC was clearly at fault, the software was developed in a way that allowed the mistake to happen without raising alerts either when the RTL altitude was incorrectly configured or during flight when the altitude violation actually occurred.

Teams building safety-critical software products are required to perform a rigorous hazard analysis [44], using techniques such as Software Fault Tree Analysis (FTA) [59, 61] or Software Failure Mode, Effects, and Criticality Analysis (FMECA) [47, 57] to identify hazardous states and a set of mitigating actions which are linked to safety-related requirements. While the safety analysis must be performed on individual products, several studies have shown that preliminary hazard analysis can be initially performed at the domain level and then contextualized during the application development process to individual products [15, 18, 31, 62]. Various frameworks, checklists, and templates exist to guide systems and software engineers through the process of identifying and mitigating hazards associated with the development and deployment of sUAS [20]. However, these tend to focus on system-level hazards while paying scant attention to the unique human interface aspects of multi-user, multi-agent systems that are emerging in the sUAS domain [2, 43, 46, 65].

Furthermore, while *human-related hazards* in the sUAS domain share commonalities with those from several other domains such as multi-agent robotics, autonomous vehicles, and drones used in the defense domain, they also exhibit unique safety concerns introduced by the deployment of remotely controlled sUAS in potentially populated areas, limited training of the remote pilots who may be ill-prepared to handle off-nominal cases, and a rapidly emergent market of sUAS applications, which in many cases are developed by hobbyists without training in safety assurance.

This paper describes domain-level safety analysis assets that explicitly focus upon human-interaction hazards. Our aim is to create a shared and *reusable resource* and a *structured process* for use by software and systems engineers working in the space of sUAS application development. We followed a systematic process that started by reviewing a broad range of academic literature and white papers describing implemented sUAS frameworks, templates, and hazard analysis associated with sUAS systems and found that the majority of hazards identified from the literature are system-oriented and fail to capture hazards associated with human-sUAS interactions. We then applied a *systematic process* that built upon the system hazards to identify additional hazards associated with *Human Interaction Points (HiPs)*. This analysis resulted in a set of domain-level hazard trees designed for safety analysis of diverse sUAS systems which we evaluated in two ways – first, against detailed accounts of publicly reported sUAS incidents, and second, through a study involving six developers with domain experience working with sUAS. Examples throughout the paper are primarily drawn from our own DroneResponse system [2, 13, 16].

The remainder of the paper is laid out as follows. Section 2 reports on our systematic process for identifying human-sUAS interaction hazards from existing literature. Section 3 describes the process we followed to construct our hazard trees, while Section 4 discusses their use. Section 5 evaluates coverage of the hazard trees against reported incidents, while Section 6 reports our study with domain experts. We analyze results in Section 7. Finally, Sections 8 to 10 discuss threats to validity, related work, and conclusions.

2 sUAS HAZARD ANALYSIS

We performed a systematic literature survey to identify an initial set of sUAS hazards based on publications reporting safety analysis techniques applied to sUAS domains. Papers covered topics such as hazard analysis, Fault Tree Analysis, and safety analysis using the Goal Structuring Notation [40]. Our aim was not to analyze the effectiveness of different techniques or frameworks but to identify specific types of hazards, faults, and safety requirements reported by authors. The resulting collection of sUAS hazard trees forms the foundation for our subsequent, more focused, and human-centered safety analysis (cf. Section 3).

2.1 Data Aggregation and Analysis Process

Our systematic literature search used the ACM, IEEE, and Scopus digital libraries to identify research papers containing descriptions of sUAS safety, failures, and incidents. We performed a number of pilot searches based on keywords collected from an initial set of papers and refined the search terms multiple times to ensure that relevant papers were part of the search including, for example, "sUAS" (and various synonyms such as UAV) and also "safety", "hazard" or "fault" analysis. Two researchers then collected and analyzed the search results, using the Parsifal tool [53]. We removed duplicates and excluded papers according to the following inclusion (IC) and exclusion (EC) criteria:

- **EC1:** Papers not related to sUAS (e.g., airplanes, or large (military-grade) UAVs) were excluded.
- **EC2:** Papers not written in English or not available as PDF via the digital library were excluded.
- **IC1:** Only papers containing information on sUAS safety-related information including safety requirements, hazards, or faults *or* papers containing information on multi-agent safety-related information with regards to interaction/collaboration, between agents (in the title and/or abstract) were included.

Our initial search returned 3462 papers. Applying the exclusion criteria resulted in a set of 2036 papers, which were reduced to 120 papers after the inclusion criteria were applied.

2.2 Hazard Tree Construction

We then skimmed each paper to identify concrete hazards, faults, and/or safety-related statements mentioned in the paper. We refer to these as *"safety statements"*. We extracted a total of 200 safety statements from 27 papers, while the remaining 93 papers did not contain specific examples of safety-related statements for sUAS.

Each of the statements was then transformed into one or more explicitly stated hazards – each one based on a safety requirement, fault, or actual hazard reported in the paper. We then used open coding to identify hazard categories, and as a result established an initial grouping of eleven different categories (e.g.,"sensors", "route planning"). This encoding process was performed by two researchers with frequent discussions and refinement of the groups and categories until agreement was reached. We finally identified eight categories of hazards. In cases where a hazard was related to multiple categories, we selected the most appropriate one. This resulted in a final set of 114 distinct "hazard statements".

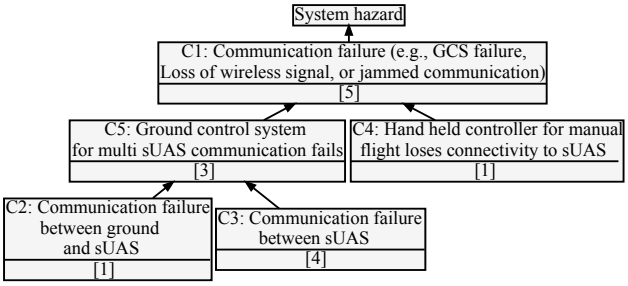


Figure 1: One of the smaller hazard trees derived from the literature survey for communication failures. The numbers indicate the number of associated hazard statements.

As the granularity of the reported hazards varied significantly, including specific hazards such as *"The sUAS deviates from its predefined route due to wind shear"* and *"Loss of satellite signal"*, we established a hierarchy of hazards (cf. Fig. 1) organized under eight hazard categories. Where hazards could be grouped under multiple parent hazards we selected the most appropriate one to avoid duplicates, but added a cross-reference to the other hazard tree when appropriate. We organized all associated hazards into a hierarchy – where necessary adding intermediate hazards, merging duplicates or very similar ones, and removing hazards that were deemed out of scope for an sUAS application or too abstract. Following this process, 108 hazard nodes remained. We then double-checked the original list to ensure that the trees provided full coverage of the hazards identified from the literature and that none had been missed. This task was performed independently by three researchers on our team. In cases where the assignment was not unanimous, we discussed the mapping until consensus was reached. This resulted in five hazards being slightly modified or newly added to the tree.

The resulting hazard trees, based on our literature search, represented use of prohibited airspace (7 hazard statements), separation distance (10 hazard statements), communication (10 hazard statements), hardware and sensor failures (37 hazard statements), weather (9 hazard statements), pilot error (15 hazard statements), preflight checks (6 hazard statements), and situational awareness (14 hazard statements).

3 HUMAN-SUAS INTERACTIONS

One of the main findings of our literature survey was that the majority of hazards and safety-related statements target system-level hazards, paying little to no attention to user-related, or user-induced human-sUAS interaction hazards that can lead to critical system failures. Given this lack of documented human-interaction hazards, we applied a systematic process for deriving them from the system hazards when performing hazard analysis for a certain use case scenario. Our process is summarized in Fig. 2. We started (step 1) by selecting a system-level hazard from an existing hazard tree. Next (step 2) we systematically explored each *Mission Mode* and identified relevant scenarios that represented *Human Interaction Points* (step 3) for each mode. Given a specific HiP, within a given mode, we systematically explored the role a human could play in instigating or mitigating the hazard (step 4) with respect to system design, user design, and hardware and configuration flaws. We

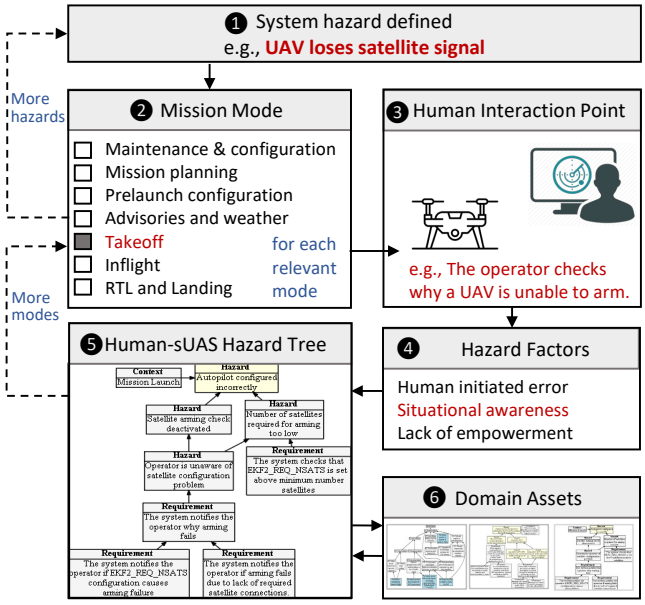


Figure 2: Hazards and mitigations with associated human factors and impacts were identified systematically for each system hazard according to mode, error type, and finally the impact upon human operators.

used this information to augment the basic system hazard trees to create a *HiP-oriented hazard tree* capturing HiP hazards for different hazard factors. We finally consolidated hazard trees for all relevant modes associated with the specific hazard (step 5). We now describe these steps in more detail.

3.1 Mission Modes (Step 2)

All sUAS systems transition through different phases of operation (i.e., modes). These include pre-flight configuration (including maintenance and mission planning), prelaunch configuration checks on sUAS readiness and retrieval of flight advisories, mission launch, (post-launch) flight, and RTL (return to launch). Hazard scenarios, including those associated with human interactions, may occur across multiple phases of operation with mode-specific characteristics and mitigations [8]. Therefore, hazard analysis must be performed for each mode [26] and must address the system's ability to detect and react to both normal and abnormal events [32]. We briefly summarize each of the modes that we include in our examples throughout the paper, as their unique characteristics shape their associated hazards and mitigations.

- **Mission planning:** (Multi-)sUAS missions are driven and constrained by mission plans that establish rules, mission boundaries, and goals, enable specific tasks and transitions between tasks, and set autonomy levels for the sUAS [13]. Actions taken in this mode can impact safety, for example if an RPIC plans a mission without considering changes in terrain elevation.
- **sUAS Maintenance and configuration:** Flight Controller systems, such as Ardupilot [5] and px4 [54], have large numbers of configurable parameters. For example, px4 lists 1,382 configuration points each with a range of allowed values. Users can configure

sUAS using third-party software packages, many of which fail to provide meaningful constraints.

• **Prelaunch configuration checks:** The system must check critical configuration values prior to launch and either correct them automatically or clearly display warnings. Common errors, as observed in our study, include setting incorrect fail-safe values (e.g., RTL altitude above the legally allowed levels).

• **Advisory and weather checks:** RPICs are required to obtain flight authorization prior to entering controlled airspaces and to check weather conditions. Incidents often involve adverse weather conditions and/or unauthorized flights into controlled airspace.

• **Takeoff:** Problems not identified during pre-launch activities can emerge at takeoff, for example, a cable that obstructs a propeller causing loss of control, communication failures between sUAS, or sUAS placement on an obstructed launch pad.

• **Inflight:** During flight, users issue directives to the sUAS. In addition, the system must provide situational awareness to users so that they can perceive the current situation, comprehend what is happening, and make sound decisions [27]. Accidents occur when users lose situational awareness, often due to well-documented design "demons" such as information overload [39, 48], attentional tunneling [56, 60, 64], and out-of-the-loop syndrome [9].

• **RTL and landing:** RTL and landing modes present unique hazards – for example the need to provide safe passage home if all sUAS simultaneously transition to RTL due to global loss-of-signal.

3.2 Human Interaction Points (Step 3)

Humans directly interact with sUAS in many different ways, including through physical manipulations (e.g., attaching a camera), use of manual flight controls (e.g., if a human takes physical control of the sUAS from the software system), as well as through issuing commands, feedback, and setting goals via the user interface [33, 63]. We discuss five common human-sUAS interaction patterns (P1-P5), based on accounts of human interactions reported in the literature (e.g., [4, 12, 33, 63]). Each of the following patterns involves actions and interactions performed by both sUAS and humans. Square brackets [] depict optional actions.

P1: **Monitor** (Human) →
[Seek_Explanation (Human) → **Explain** (sUAS) →**]**
[**Intervene** (Human) → **Respond** (sUAS)]

P2: **Request_feedback** (sUAS) → **Provide_feedback** (Human) → **Act** (sUAS) → **Monitor** (Human)

P3: **Adapt+Explain** (sUAS) → **Monitor** (Human)

P4: **Observe and Configure** (Human) →
Check_Configuration (sUAS or System)

P5: **Set_mission_goals** (Human) → **Plan_mission** (System+sUAS) → **Act** (sUAS)

The first pattern (P1) is the most common one in a HoTL system where a human operator's responsibility is primarily supervisory. The operator monitors the system and when needed requests an explanation from the sUAS system. The human may decide to intervene in the sUASs' behavior, and the sUAS responds accordingly. In pattern P2, an sUAS explicitly requests permission to perform an action or requests confirmation that a taken decision was correct. An example of this is when an sUAS uses onboard vision to detect a victim during a search-and-rescue mission, and requests confirmation from the operator that it made a correct decision to switch from "search" to "track" mode [2]. The operator provides feedback which the sUAS acts upon, and the human monitors the response.

In P3, the sUAS adapts independently and then provides an explanation of the adaptation. An example would be when an sUAS makes an RTL decision due to low battery. The human monitors the action and if necessary intervenes by following pattern P1. In P4, the human configures and checks the system – either during flight or prior to flight. Finally, in P5, the human sets mission goals which guide and constrain the actions the sUAS are allowed to take including tasks they will perform, permissions to act autonomously, and ways in which they will collaborate with other sUAS and with human supervisors. We explored instances of these patterns across common and exception scenarios for diverse flight modes, and used them to aid in the identification of human-interaction points.

3.3 Human Hazard Factors (Step 4)

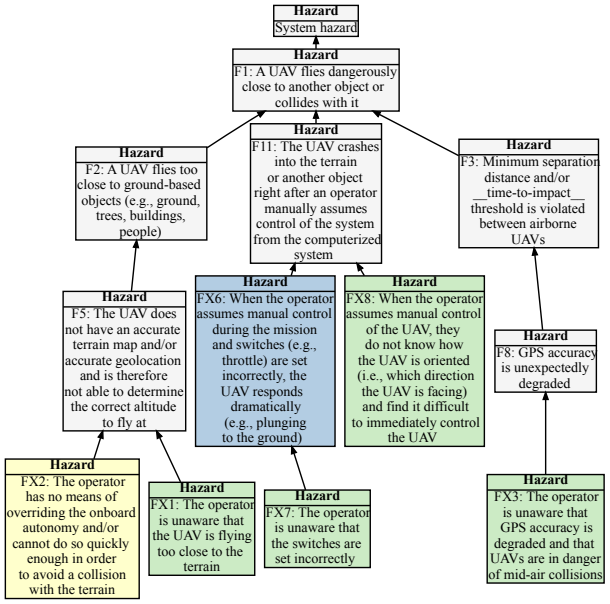
We further explore three types of human-interaction errors. *Human initiated errors* are quite common in the sUAS domain, as many pilots have limited training. Examples include ignoring regulations and restrictions, or failing to follow established processes. *Loss of situational awareness*, occurs when the user is unable to fully perceive the current situation, comprehend what is happening, and make sound decisions [27]. For example, if the system provides inaccurate information about the health or location of an sUAS, or fails to explain why an sUAS behaves in a certain way (e.g., the sUAS stopped at a certain point and does not move further), the user may make suboptimal decisions for how to proceed with the mission. Finally, *lack of empowerment* occurs when the operator is aware of the state of the mission, knows what they would like to do, but the system does not provide the means for them to do it. A simple example is when the system fails to provide the user with the option of canceling a flight route currently in progress.

3.4 Constructing the Hazard Trees (Steps 5 & 6)

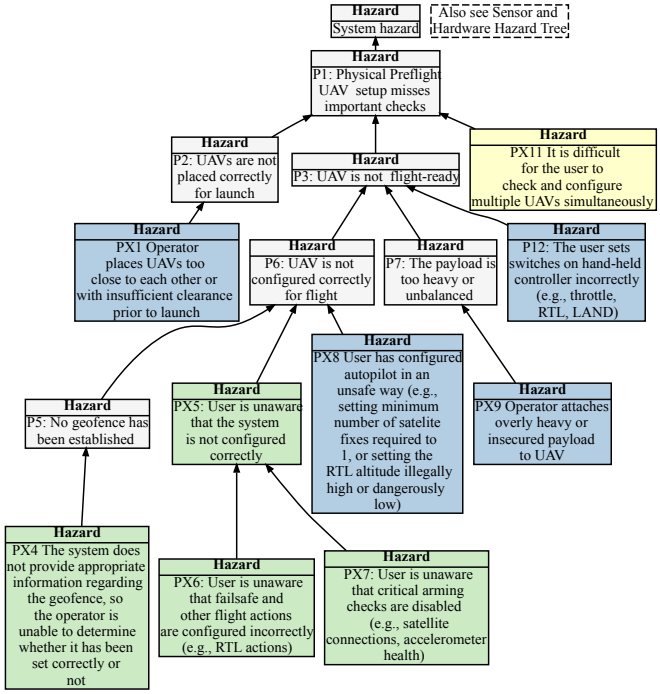
Finally, based on this systematic approach for identifying human-sUAS hazards, we constructed hazard trees capturing the system hazard, sub-hazards, and human-sUAS hazards to be addressed. Partial examples for two hazard trees (sUAS collisions and pre-flight configurations) are reported in Fig. 3. These initial trees were continually refined into domain-level assets (step 6) throughout the remainder of our study as additional hazards emerged. This refinement process is further described in Section 5.2 based on additional hazards discovered through analyzing reported incidents. Our final set of eight hazard trees is listed in Table 1. We also identify candidate mitigations. For example, in Fig. 3b, hazard PX7 could be mitigated by the requirement that *"The system shall store a list of default arming checks to be applied to all UAVs by type (e.g., PX4, Ardupilot). An alert shall be displayed if any UAV's internal configuration differs from the expected arming checks."*.

While this paper does not focus on the process of establishing mitigations, our GitHub-hosted hazard trees do include a (growing) list of mitigation options for each human-interaction error[1].

[1] sUAS Repository: https://github.com/SAREC-Lab/sUAS-UseCases



(a) Hazards related to sUAS collisions



(b) Hazards associated with preflight checks and configurations.

Figure 3: Two partial hazard trees. System nodes were derived from the literature survey, while colored nodes were derived from our analysis of human-interaction errors. Legend: Gray=system hazards, blue=human initiated errors, green=loss of situational awareness, yellow=lack of empowerment.

Description: Multiple UAVs dispatched to search for victim.
Primary Actor: Drone Commander (DC)
Trigger: The DC activates the search.

Main Success Scenario:

1. The UAV performs synchronized takeoff
[mode: Takeoff]
a) UAVs are not placed correctly for launch (system haz.)
i.Operator places UAVs too close to each other for launch [HE]
ii.Operator places UAVs in area with insufficient clearance [HE]
....

2. DroneRescue tracks and displays the location and state of each UAV [mode: Inflight]
a) Communication Failure between ground and UAVs (system haz.)
i. The operator is unable to receive status data from the UAVs and loses situational awareness [SA]
...

(more steps….)

Figure 4: Use case Vignette includes (system-level hazards), mission modes, and (respective HiPs), where HE=Human Error, SA=Loss of Situational Awareness.

4 LEVERAGING THE HAZARD TREES

In this section we assume the role of an end user (e.g., an sUAS system developer) and show how a user could leverage our hazard trees, and the process we developed, to identify relevant human-interaction factors for a specific sUAS application. Our example is based on the use case vignette shown in Fig. 4 for a search-and-rescue scenario. We systematically examine each step of the use case and identify its HiPs and their associated flight modes and hazard groups. In this example we focus on the use case step related to *synchronized takeoff* which occurs in *takeoff mode*. We identify a HiP in which the operator prepares the sUAS for launch, and then select relevant hazard trees of *preflight configuration, weather, and mission planning*. The user retrieves those hazard trees and utilizes them to aid in the hazard analysis process. In this case, we identify our first system-level hazard from the preflight configuration tree stating that "UAVs are not placed correctly for launch". The tree offers two associated HiPs defined as *human errors* for "Operator places UAVs too close to each other prior to launch" and "Operator places UAVs in location with insufficient clearance prior to launch". We follow a similar process for all subsequent steps.

Once human-interaction hazards are identified, we can propose mitigating requirements. For example, the inappropriate placement of sUAS could be mitigated through including a clearance check in a prelaunch checklist, or by adding a new feature to the system that raises a placement alert if the minimum separation distance is violated between sUAS prior to launch.

5 INCIDENT REPORT COVERAGE

The first part of our evaluation assesses the *coverage of human-sUAS incidents across the hazard trees*. We collected reports of incidents, accidents, and failures related to sUAS usage from news services

Table 1: Final, refined Hazard Groups with system- (SYS) and user-hazards human-sUAS interaction hazards (HI)

ID	Cause of Hazard	Hazards SYS	HI
CL	**Collisions:** between sUAS, other objects, and terrain	9	7
CM	**Communication:** loss of comm with sUAS	7	7
HS	**Hardware/Sensors:** e.g., cameras, GPS, parachutes	14	7
MA	**Mission Awareness:** Mission status, decision making	7	11
MP	**Mission Planning:** Flight routes and task allocation	5	3
PC	**Preflight Configuration:** Geofence, launch, params	7	12
RC	**Regulatory Compliance:** Airspace, flight constraints	9	9
WT	**Weather:** Extreme temperature & wind	4	9

Table 2: sUAS Incidents involving as reported publicly and/or through regulatory bodies

Source	URL	#
Aviation Safety Reporting System (ASRS)	http://asrs.arc.nasa.gov/docs/rpsts/uav.pdf (ACN: 1599671)	50
Wikipedia collection of incidents	en.wikipedia.org/wiki/List_of_UAV-related_incidents	109
dedrone – Collection of Worldwide Drone Incidents	www.dedrone.com/resources/incidents/all	100
The Center for the Study of the Drone – Bard College	http://dronecenter.bard.edu/drone-incidents	30
UK Air Accidents Investigation Branch reports	https://www.gov.uk/aaib-reports?keywords=UAS	50

and regulatory bodies. In total we inspected 339 reports of sUAS incidents from five different sources as depicted in Table 2. Three members of our team analyzed the incidents and mapped reported human-interaction failures to the eight hazard groups and to the three hazard factors. For each incident, one team member performed the initial mapping and a second member checked the mappings. In case of disagreement, all three people discussed the results to reach consensus. Aggregated results are reported in Fig. 5.

5.1 Results

The majority of reports simply stated that an sUAS was sighted in prohibited airspace without any discussion of the contributing cause, while only 54 provided detailed accounts. 43 of these included human-related factors, providing insights into the prevalence and root causes of different incidents and indicating that the majority of incidents involved human factors. For example, some incidents related to entering a prohibited airspace were caused by lack of preflight configuration or appropriate configuration checks (covered by the hazard trees "Regulatory Compliance" and "Preflight Configuration"). These accidents could be avoided by using a flight authorization system. Similarly, in the case of pilot errors (e.g., losing line-of-sight to the sUAS), or hardware errors, (e.g., loss of GPS), appropriate prelaunch configuration checks such as correct fail-safe settings, could have mitigated these incidents (cf. "Preflight Configuration").

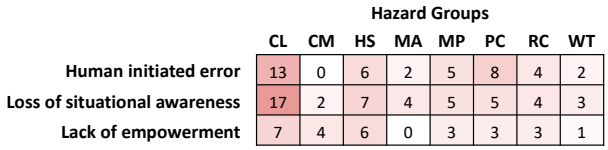


Figure 5: A heatmap showing human-interaction factors mapped against hazard groups according to their occurrence in the analyzed incident reports.

Three of these incidents are summarized in Table 3 [I1-I3]. Two cases (I1,I3) were related to (near) collisions with other aircraft, and all of them involved *human initiated error* either *inflight* (I1, I3) or as a result of *preflight configuration* errors (I2). Human initiated errors included ignoring airspace warnings (I1), flying BVLOS (beyond visual line of sight) (I1), and failing to obtain permission to fly in controlled airspace (I3), both of which could well be mitigated through imposing constraints on flight planning (see hazard tree *Regulatory Compliance*) and increasing situational awareness levels through providing more warnings, recommendations, and even prohibitions (see tree *Mission Awareness*). In the second incident (I2), the RPIC illegally set the RTL altitude to approximately 750 feet in order to avoid pylons and other obstacles if a fail-safe caused the sUAS to switch to RTL mode. As sUAS can be configured using open-source applications (e.g., QGroundControl [55] or MissionPlanner [6]), their configurations must be checked for undesirable settings immediately prior to flight. Responsible software packages should also warn anytime a user configured the autopilot in a potentially illegal or dangerous way. Finally, in incident I3, the operator reported that he suffered from stress due to multitasking, and failed to notice that the system had started reporting altitude in meters and not feet. This type of stress is quite common when humans supervise autonomous systems, and is referred to as Workload, Anxiety, Fatigue, and Other Stressors (WAFOS) by Endsley [27]. However, the incident could have been avoided with a legally established geofence or by raising alerts if unexpected configurations were introduced. These three incidents highlight the importance of software engineers designing, implementing, and testing sUAS systems systematically to address safety concerns related to human-sUAS interactions.

5.2 Analysis & Hazard Tree Refinement

As depicted in Fig. 5, many of the reported incidents included collisions and were attributed to loss of situational awareness (17 instances) or directly related to human error (13 instances). The second most common category was preflight configuration issues, where either the pilot was partially at fault (8 instances) or the system did not provide adequate or sufficient information to configure the system correctly (5 instances). Altogether, we conclude that incidents were reported for each hazard group and that all reported human-interaction factors were successfully mapped to one or more of the hazard groups.

Based on our observations we performed a few updates to improve the structure and clarity of the trees. We observed that pilot/operator related hazards occurred primarily due to loss of mission awareness and prelaunch configuration problems, and therefore redistributed pilot hazards to these two categories. We further

Table 3: Example sUAS-Incidents with Human Contributing Factors

ID	Incident Type	Incident Description	Human-Related Hazard	Ref
I1	UAV collision with hot-air balloon flying in unauth. airspace over Boise, Idaho in 2018	The amateur pilot overrode warnings about flying in unauthorized airspace close to airport without permission from ATC. The UAV went beyond visual line of sight and RPIC was unaware that the UAV was repeatedly shearing against the balloon until propellers fell off. Pilot was unskilled with hand-held controls.	Ignoring critical warnings; Flying BVLOS.	[37]
I2	Near collision with Highway Patrol helicopter over Martinez, California between 700 and 800 feet in 2015	Max. altitude allowed for UAV flights in the USA is 400ft (or 100ft above buildings). The RPIC had overridden the altitude at which UAVs return to launch to avoid electrical pylons. When signal was lost with the UAV during flight, it switched to RTL mode, and operated on autopilot at prohibited altitudes placing it into the flight path of the helicopter.	Critical default values overridden by user in a 3rd party tool; Failure to check configurations prior to launch.	[11]
I3	UAV was flown to an altitude that was in excess of the 400 FT AGL limitation specified within FAR Part 107	The RPIC believed that altitude was being reported in feet; and was not aware that it had been reset to meters. As a result he accidentally flew to approx 492 feet, claiming that the mistake was caused by his focus on avoiding flying near obstacles or over people, coupled with the delayed awareness that the software had reset to metric units.	Delayed awareness of UAV status, WAFOS (Workload, Anxiety, Fatigue, and other Stressors situational awareness demon)	[7]

extracted several hazards from across multiple categories into a new category named "Inflight Mission Awareness" which grouped hazards related to situational awareness. This new category includes hazards such as "The operator is overwhelmed by status information for multiple UAVs" and " The operator is unable to handle multiple alerts simultaneously". This produced the current set of eight hazard groups which are listed in Table 1 and were used for the subsequent study with domain experts.

6 EVALUATION BY DOMAIN EXPERTS

The second part of our study was designed to evaluate whether the hazard trees were useful for analyzing and identifying requirements associated with human-sUAS interactions. We invited six experts from the sUAS domain to review use cases describing sUAS missions and to utilize our hazard trees to augment a set of use cases with human-interaction hazards using the process described in Section 4. None of these participants were involved in the development of the hazard trees or in the development of the systematic process.

6.1 Study Design

The study was divided into three parts, that included an initial semi-structured interview, an analysis task, and closing interview. All phases of the study focused on two primary use cases in which sUAS were used to support (1) river search and rescue, and (2) environmental water sampling. Both top-level use cases invoked supporting use cases to *activate and arm* sUAS, *generate flight routes*, and to plan non-intersecting routes through *leasing airspace*. In addition, the river rescue use case invoked *active victim tracking* and the environmental sampling use case invoked *flight authorization*. We piloted the study internally with one user, made improvements, and then conducted the study as follows.

Initial Briefing: At the start of the interview we described the Hazard Factors that we had addressed, namely *human initiated error*, *lack of situational awareness*, and *lack of empowerment*; however, we also stated that discussion was not limited to these factors. We then presented participants with one of the primary use cases and asked them to brainstorm potential safety hazards for each step of the use case with the focus on human interaction and/or human failures using a think-out-loud protocol [36]. We time-boxed this discussion to 15 minutes, as that was sufficient to understand the *types of issue* each participant would identify, while seeking a complete analysis would require much longer. Our aim was to establish a baseline for how developers currently think about, and identify human-sUAS hazards. The interview was recorded using Zoom and automatically transcribed for later analysis.

After the brainstorming task was completed, we spent approximately 10 minutes explaining the task to be performed. We introduced the eight hazard groups and their associated hierarchies of system and human-interaction hazards and then pointed the participants to an 8 minute (take-home) online video that they could use to further familiarize them with the hazard trees and the study process.

Study Task: The study task was performed individually. We assigned participants into one of two groups (river rescue, environmental sampling). Each participant was given their own multi-sheet google spreadsheet which included (1) a summary of the 8 hazard groups, (2) individual sheets for the primary use case, and four supporting ones. Each use case included metadata, a list of success steps, and a matrix for evaluating each step and marking (a) whether each of the 8 hazards was relevant, and (b) in the case of supporting use cases, exactly which hazards from the hazard trees were relevant. Fig. 6 shows the main part of the spreadsheet for one particular supporting use case. Hyperlinks allowed the user

	Supporting Use Case: Active Tracking	Hazard Group								Specific hazard mappings
		CL	CM	HS	MA	MP	PC	RG	WT	
1	The UAV positions itself in tracking_position i.e., near the victim but not directly overhead.	☑	☐	☑	☐	☐	☐	☐	☑	W2, F4, S2, S4, S7, S10, S11, S14
2	The UAV uses image capture and analysis to continually tag the victim in the image stream.	☐	☐	☑	☐	☐	☐	☐	☑	S7, S13
3	Onboard tracker continually calculates the relative position of the victim with respect to the UAV.	☐	☐	☑	☐	☐	☐	☐	☐	S11, S12, S7
4	The onboard tracker generates appropriate velocity vectors to fly towards the victim...	☑	☐	☑	☐	☐	☐	☐	☑	S11, S12, S7
5	Velocity vector is sent to the UAV's autopilot and executed to enable the UAV to track the victim.	☐	☑	☑	☐	☐	☐	☐	☐	S7
6	Steps 2-5 are repeated until the UAV receives a [stop_tracking] command.	☐	☐	☐	☐	☐	☐	☐	☐	

Figure 6: An example of the hazards identified by one of the participants for the supporting use case of Active Tracking.

to easily move between use cases and to view the hazard trees on GitHub We instructed participants to spend up to 45 minutes, to systematically evaluate each use case step and to mark the relevant hazards.

Follow-up Interview: In a follow-up interview we asked participants a series of open-ended questions about their experience in working with the hazard trees. Questions included (1) To what extent did the trees help you to identify potential human interaction errors? (2) Was it difficult to find a matching hazard group and/or specific hazard for a use case step? (3) Was there anything missing or unclear with the groups or hazard trees? (4) When flying a drone, have you ever experienced an incident that was at least partially caused by a human-sUAS interaction problem? (5) If so, do you think that incident could have been prevented or mitigated if you had addressed hazards described in the hazard trees (please explain your answer)? In addition, we used a rubric to elicit feedback on usability and efficiency of the approach. The follow-up interview was also recorded and transcribed.

6.2 Study Participants

We recruited six participants for our study, each of them with extensive domain expertise in piloting sUAS or developing non-trivial sUAS applications. As shown in Table 4, their experience ranged from 2 to 8 years working on sUAS development projects, with an average of 3.67 years. We also indicate whether they had experience with software, hardware, and/or in a multi-sUAS environment. In cases where a participant had prior experience with river rescue (P3) or environmental sampling (P2, P6), we assigned them to those respective use cases. We opted to include only highly qualified domain experts who represent our target users, even though this reduced the size of our participant pool. However, Nielsen [51] has shown, that five or six participants are sufficient for providing meaningful and in-depth feedback on a research design solution such as the hazard trees.

6.3 Initial Interview Analysis

We performed an inductive coding analysis [45] on the transcripts from the initial meetings. The researcher that conducted the interview performed the initial analysis and this was then cross-checked by a second researcher. We identified four themes that were observed across several of the discussions. First, several of our participants *focused upon system-level hazards* rather than human-related ones, and were sometimes unable to identify any human-interaction hazards. For example, participant P4 stated that "*I don't see any opportunities for human error*" associated with defining a coverage area and allocating routes to sUAS. In fact, five out of six of our participants (with the exception of P2) focused more on system-level hazards than human-related ones. In a closely related theme, three of the participants requested additional explanations about human-interaction errors, indicating that this was a new concept for them. Finally, we observed several examples of *"blaming the operator"* without consideration for how a design flaw in the system might have increased the likelihood of an operator error. For example, P5 observed that the user could set an incorrect mode for takeoff (human error), but did not mention that the system failed to raise an alert which could have notified the operator of the problem.

Table 4: Study participants

#	Application domain	Exp	UC	Mult	HW	SW
P1	sUAS dispatch & Call center UI	4 yrs	ES		•	•
P2	Environmental Applications	8 yrs	ES		•	•
P3	Multi-sUAS Search and Rescue	2 yrs	RR	•		•
P4	Safety& Security/Emerg. Resp.	2 yrs	RR		•	
P5	Defibrillator Delivery	4 yrs	RR		•	•
P6	Environmental Applications	2 yrs	ES		•	•

Legend: ES=Environmental Sampling, RR=River rescue, Mult=Multi-sUAS development, HW=Hardware, SW=Software.

6.4 Task Analysis and Follow-up Interviews

Once participants had completed the task, we analyzed the mappings from use case steps to hazards. We report aggregated results in Table 5. For the two primary use cases (River Rescue and Environmental Sampling), most participants found mappings to all eight hazard groups. For the supporting use cases, agreement was significantly lower. Out of 48 potential mappings (i.e., 6 supporting use cases × 8 hazard groups) there were only seven cases in which all participants agreed to the mapping, but 15 cases in which at least 2/3 agreed. Similarly, there were eight cases of full agreement that the hazard group was not relevant and 20 cases with at least 2/3 agreement. There were only five potential mappings which lacked 2/3 consensus.

As reported in Table 5, the participants found each hazard group to be useful across at least some of the use cases. In total, Communication (or the loss thereof) was mentioned 58 times in total, Hardware/Sensors were mentioned 52 times, and Mission Awareness mentioned 51 times. Collision was mentioned the least with only 24 mentions over all use cases. All six participants mentioned the

Table 5: Participant majority consensus for whether a specific hazard group was relevant for two top level use-cases (River Rescue and Environmental Sampling) and five supporting use cases.

	Hazard Group							
Use Case	CL	CM	HS	MA	MP	PC	RG	WT
River Rescue	●	●	●	●	●	●	●	●
Environmental Sampling	●	●	●	●	●	●	●	●
Activate & arm	○	●	●	○	○	●	○	○
Area coverage	○	○	○	◑	●	○	●	○
Flight authorization	○	○	○	●	●	●	●	●
Lease airspace	○	●	◑	●	◑	○	●	○
Active tracking	◑	●	●	○	○	○	◑	○

Legend: ●: 2/3 Majority Agreement that the hazard group is related, ◑: Lack of Agreement, ○: 2/3 Majority Agreement that the hazard group is not related. The highest ranked hazard groups per use case are underlined in red.

usefulness of the hazard trees. For example, P2 stated that *"the trees were very useful way to connect the hazards"*, while P3 stated that *"they were super useful...there were things I wouldn't have thought of"*. We also observed several cases in which a participant had stated during the initial interview that there were no human-interaction hazards associated with a specific use case step, but found relevant ones when using the hazard trees. For example, when equipped with the hazard tree, P3 identified previously undetected hazards associated with "Loss of Communication", "Mission Planning", and "Regulatory Compliance" for the area coverage use case. This indicates that the hazard groups provide valuable information regarding human-sUAS interactions.

Based on a 4-point Likert scale (very efficient, efficient, inefficient, very inefficient), 4 out of 6 participants stated that use of the hazard groups was *"efficient"*, and that the grouping improved the assignment task; however, two participants rated the task of identifying and assigning hazards as inefficient. Subsequently, the feedback was somewhat mixed. P2 stated that he *"was hunting around a bit"* but that he *"liked the categorization"* and "the color coding" while P5 stated that *"the organization of hazards within each of the respective trees makes it pretty easy to search a tree"*. P3 shared that *"After 15 or 20 minutes everything seemed to kind of make sense"* but that *"there is a lot of information because it is a complex problem"*. We discuss this issue further in Section 7.

Several participants mentioned that they leveraged systems hazards to help them identify human-interaction hazards. P4 explained that he first *"identified system flaws and then followed them down the tree to uncover potential human-sUAS interaction problems"* while P5 stated that he *"started at mid-level (system) hazards and found relevant human-interaction hazards below that"*.

Finally, all participants were able to elucidate on at least one human-interaction problem they had personally experienced, and all agreed that knowledge of the hazard trees during the development process could have helped them foresee and mitigate the reported problem. For example, P1 described a real-life incident he had experienced which led to unnecessary human intervention resulting in a crash, and stated that *"we probably would have thought more about (how to address the hazards) had we looked over those hazard trees"* and said that *"preflight check(s) could have alerted to the throttle position"* thereby preventing the accident. In general, our participants indicated that the real value of the trees is in providing examples that encourage *safety thinking*.

7 DISCUSSION

Based on the analysis of reported incidents and sUAS accidents by regulators and the media, we found that human initiated errors and loss of situational awareness dominated the reported incidents. Based on our analysis and the feedback from our domain experts, we conclude that several of the reported incidents could have potentially been prevented or mitigated by addressing the hazards collected in our domain-level trees.

Our study with domain experts indicated that developers in our study focused on system-level hazards, most likely because they were more familiar with addressing system problems and less knowledgeable about the role of human-related hazards. This was the case even though they understood the full scope of the study, including its focus on human-interaction errors. Based on the mappings they created when using our hazard trees and their feedback in the final interview, we conclude that the hazard trees facilitated safety thinking from a human-interaction point of view and provided a good starting point for "digging deeper" into these types of hazards.

A second important consideration is the need for better tool support. While our study participants reported that identifying types of hazards (aka hazard groups) was quite easy, finding specific hazards added an additional level of complexity and as a result, sifting through all hazards in all (relevant) trees was rather tedious. This led to the conclusion that additional tool support could ease the burden of looking for potential matching hazards. While we do not propose a "checklist-like tool", as this could provide a false sense of completeness, features for searching, filtering, and annotating the trees could ease the task of identifying relevant hazards.

Table 6: Two human-interaction hazards with examples of potential mitigations.

Hazard: *PX1: Operator places UAVs too close to each other prior to launch*
PX1-S1: RPIC receives proper training to conduct mandatory preflight checks (Process Requirement).
PX1-S2:The system checks the coordinates of all UAVs on the ground and raises an alert if any of them are located less than minimum separation distance apart (Safety Requirement).
Hazard: *CX3: The human operator is unable to receive status data from the UAV using the software-based system.*
CX3-S1: The approximate position and the uncertainty of the UAV's current position on the map must be visually depicted (e.g., by creating an increasingly large "circle" around the last known, or projected position of the UAV (Safety Requirement).

In addition, our aim is to provide a resource for addressing hazard analysis and safety assurance throughout the software engineering lifecycle in order to aid sUAS developers in building safer systems that empower and support diverse operators. Our hazard trees include a set of candidate mitigations associated with each human-interaction hazard as illustrated in Table 6. The current list of mitigations is included our online repository. Our approach is designed to address an emergent problem in the domain of sUAS development by providing a reusable set of hazard descriptions and mitigations aimed at inspiring and supporting a safety mindset for sUAS developers.

8 THREATS TO VALIDITY

Our work is subject to several validity threats. While we have shown that it is applicable to real-world incidents and use case scenarios, the limited number of incident reports makes it likely that other types of incidents are not yet covered. Furthermore, additional external evaluation of the process is required to ensure its applicability in a more broader scenario. The analyzed incident reports sparsely cover multi-sUAS applications and custom software solutions. The majority of reports are related to the use of off-the-shelf applications (such as Mission Planner and DJI's propriatery software system), and do not represent multi-sUAS missions. Given the increasingly common reports of sUAS incidents and the dearth of information discussing root causes, we have created a shareable resource that can be used as a starting point for analyzing human-sUAS hazards in a specific application.

We considered several alternate study designs, including a controlled experiment that would involve specifying requirements for a system with, and without, our hazard trees. However, to accomplish this in a non-trivial way would require significant time and effort, beyond available resources. Our approach, falls under the broad umbrella of design science, in which it has been shown that even a limited number of participants provides useful usability feedback to iteratively refine a design [51]. Another threat is related to the experience of the participants. While all participants have experience in handling and operating sUAS, only one participant had previous experience with multi-sUAS missions. Therefore, we expect additional human-sUAS interaction hazards to emerge from these types of systems and application use cases. We release our set of hazard trees as a publicly available community resource that is meant to evolve over as new incidents are reported from which hazards can be identified.

We attempted to minimize internal validity threats associated with the incident analysis and study, by dividing the transcription of audio recordings among two researchers and cross-checking the resulting transcripts, codes, and emergent themes. For the incident coverage, three researchers performed the analysis and each incident was checked by two researchers. In case of a disagreement, the incident was reevaluated until agreement was reached.

9 RELATED WORK

The most closely related work is in safety assurance, hazard slicing, and situational awareness.

Safety Assurance: Work by Denney and Pai [19, 21, 24, 25] in this area addresses different aspects of sUAS and UAV safety providing automation support and tools for creating and maintaining safety assurance cases. They emphasise reuse of safety assurance cases by proposing domain-independent and domain-specific patterns [22, 29]. Other work has created reusable safety case patterns as building blocks for future product development [10, 19, 21, 23, 29, 35]. In the area of safety cases maintenance, Kelly and Weaver [40] presented a set of patterns and recommended the use of modularity to support safety cases evolution. Kelly and McDermid [41] investigated changes in evidence, context, assumption and requirements nodes to determine how changes impact the safety assurance case.

Hazard Slicing: Similar to our approach of dividing a large hazard tree into several sub-trees to better address the different aspects, several researchers have shown the benefits of hazard-based slicing [38, 58]. Agrawal et al. [3, 14] proposed SAFA (Software Artifact Forest Analysis) that uses underlying trace links to create and visualize hazards trees and their respective mitigations in the context of an evolving safety-critical software system. SafeSlice [28] extracts design slices based on functional safety requirements. Other work in this area focuses on generating artifact slices using formal verification techniques to support safety analysis [38, 58]. However, these approaches are all system focused with little to no emphasis on user induced hazards and faults. While these types of hazards are important, specific processes, methods, or guidelines for identifying and mitigating human-interaction related hazards are missing.

HCI/Situational Awareness: The seminal work on Situational Awareness (SA) by Endsley [27] focused on user-centered design identifying eight common design errors that occur frequently in user interface designs and which inhibit SA. Several studies in this area explicitly explore SA and shortcomings of user interfaces for various types of systems such as tsunami early warning systems [1], electric mining shovels [52] or operator interactions with a single robot or machine [30, 34]. While the HCI community has examined this problem from various angles, they focus primarily on the user-interface design and not on broader sets of hazards which our approach is designed to address.

10 CONCLUSION

In this paper, we have presented an approach for systematically deriving human-interaction hazards for sUAS systems. Based on a literature survey we identified eight different categories of hazards, that serve as a starting point for a human-centered hazard analysis. As part of the process, we identified different mission modes and contributing hazard factors, derived patterns for Human Interaction Points (HiPs), and constructed an initial set of reusable hazard trees. As part of our future work we plan on further extending our hazard tree library, exploring ways to make it more scalable, including providing tool support to facilitate navigation and identification of relevant parts of a hazard tree for domain-specific multi-sUAS applications. We also plan to extend our work into the design, implementation, and test lifecycle to provide support for realizing hazard mitigations.

ACKNOWLEDGMENTS

This project has been partially funded by the Linz Institute of Technology, the US National Science Foundation (SHF-1909007, CNS-1931962) and by support from Northrop Grumman.

REFERENCES

[1] 2009. Interaction Design for Situation Awareness-Eyetracking and Heuristics for Control Centers.

[2] A. Agrawal, S. Abraham, B. Burger, C. Christine, L. Fraser, J. Hoeksema, S. Hwang, E. Travnik, S. Kumar, W. Scheirer, J. Cleland-Huang, M. Vierhauser, R. Bauer, and S. Cox. 2020. The Next Generation of Human-Drone Partnerships: Co-Designing an Emergency Response System. In *Proc. of the 2020 Conf. on Human Factors in Computing Systems*.

[3] A. Agrawal, S.Khoshmanesh, M. Vierhauser, M. Rahimi, J. Cleland-Huang, and R. R. Lutz. 2019. Leveraging artifact trees to evolve and reuse safety cases. In *Proc. of the 41st Int'l Conf. on Software Engineering*. 1222–1233.

[4] Ankit Agrawal, Jan-Philipp Steghöfer, and Jane Cleland-Huang. 2020. Model-Driven Requirements for Humans-on-the-Loop Multi-UAV Missions. In *Proc. of the 10th Int'l Model-Driven Requirements Engineering WS*.

[5] Ardupilot. 2020. Ardupilot – open source autopilot software. https://ardupilot.org. [Last accessed 01-06-2021].

[6] Ardupilot. 2020. MissionPlanner. https://ardupilot.org/planner. [Last accessed 01-06-2021].

[7] Aviation Safety Reporting System. 2021. ASRS Database Report Set: Unmanned Aerial Vehicle (UAV) Reports (ACN: 1599671). https://asrs.arc.nasa.gov/docs/rpsts/uav.pdf [Last accessed 01-06-2021].

[8] Paul Baybutt. 2012. Process hazard analysis for phases of operation in the process life cycle. *Process Safety Progress* 31, 3 (2012), 279–281.

[9] Francesco N Biondi, Monika Lohani, Rachel Hopman, Sydney Mills, Joel M Cooper, and David L Strayer. 2018. 80 MPH and out-of-the-loop: Effects of real-world semi-automated driving on driver workload and arousal. In *Proc. of the Human Factors and Ergonomics Society Annual Meeting*, Vol. 62. SAGE Publications, 1878–1882.

[10] Robin E. Bloomfield and Kateryna Netkachova. 2014. Building Blocks for Assurance Cases. In *Proc. of the 25th IEEE Int'l Symp. on Software Reliability Engineering Workshops*. 186–191.

[11] CBS SF Bay Area News Outlet. 2015. Pilot Of Drone That Nearly Hit CHP Helicopter Says It Was On Autopilot. https://sanfrancisco.cbslocal.com/2015/12/17/drone-near-miss-chp-helicopter-martinez-owen-ouyang-apology-autopilot. [Last accessed 01-06-2021].

[12] Jane Cleland-Huang and Ankit Agrawal. 2020. Human-Drone Interactions with Semi-Autonomous Cohorts of Collaborating Drones. In *Proc. of the Interdisciplinary WS on Human-Drone Interaction; co-located with the 2020 ACM CHI Conf. on Human Factors in Computing Systems*.

[13] Jane Cleland-Huang, Ankit Agrawal, Md Nafee Al Islam, Eric Tsai, Maxime Van Speybroeck, and Michael Vierhauser. 2020. Requirements-Driven Configuration of Emergency Response Missions with Small Aerial Vehicles. In *Proc. of the 24th ACM Conf. on Systems and Software Product Lines*. 1–12.

[14] J. Cleland-Huang, A. Agrawal, M. Vierhauser, and C. Mayr-Dorn. 2021. Visualizing Change in Agile Safety-Critical Systems. *IEEE Software* 38, 03 (May 2021), 43–51.

[15] Jane Cleland-Huang, Mats Per Erik Heimdahl, Jane Huffman Hayes, Robyn R. Lutz, and Patrick Maeder. 2012. Trace Queries for Safety Requirements in High Assurance Systems. In *Proc. of the Int'l Working Conf. on Requirements Engineering: Foundation for Software Quality*. 179–193.

[16] Jane Cleland-Huang, Michael Vierhauser, and Sean Bayley. 2018. Dronology: an incubator for cyber-physical systems research. In *Proc. of the 40th Int'l Conf. on Software Engineering: New Ideas and Emerging Results*. 109–112.

[17] Nancy J. Cook. 2007. *Stories of Modern Technology Failures and Cognitive Engineering Successes*. CRC Press, 2007.

[18] Josh Dehlinger and Robyn R. Lutz. 2006. PLFaultCAT: A Product-Line Software Fault Tree Analysis Tool. *Autom. Softw. Eng.* 13, 1 (2006), 169–193.

[19] Ewen Denney and Ganesh Pai. 2014. Automating the assembly of aviation safety cases. *IEEE Transactions on Reliability* 63, 4 (2014), 830–849.

[20] Ewen Denney and Ganesh Pai. 2015. Argument-based airworthiness assurance of small UAS. In *Proc. of the 2015 IEEE/AIAA 34th Digital Avionics Systems Conf.* IEEE, 5E4–1.

[21] Ewen Denney and Ganesh Pai. 2016. Composition of safety argument patterns. In *Proc. of the Int'l Conf. on Computer Safety, Reliability, and Security*. Springer, 51–63.

[22] Ewen Denney and Ganesh Pai. 2016. Safety considerations for UAS ground-based detect and avoid. In *Proc. of the IEEE/AIAA 35th Digital Avionics Systems Conf.* IEEE, 1–10.

[23] Ewen Denney, Ganesh Pai, and Josef Pohl. 2012. Heterogeneous Aviation Safety Cases: Integrating the Formal and the Non-formal. In *Proc. of the 17th IEEE Int'l Conf. on Engineering of Complex Computer Systems*. 199–208.

[24] Ewen Denney, Ganesh J. Pai, and Ibrahim Habli. 2015. Dynamic Safety Cases for Through-Life Safety Assurance. In *Proc. of the 37th IEEE/ACM Int'l Conf. on Software Engineering*. 587–590.

[25] Ewen Denney, Ganesh J. Pai, and Iain Whiteside. 2017. Modeling the Safety Architecture of UAS Flight Operations. In *Proc. of the 2017 Int'l Conf. on Computer Safety, Reliability, and Security*.

[26] Homayoon Dezfuli, Allan Benjamin, Christopher Everett, Martin Feather, Peter Rutledge, Dev Sen, and Robert Youngblood. 2015. NASA System Safety Handbook. Volume 2: System Safety Concepts, Guidelines, and Implementation Examples. (2015).

[27] Mica R. Endsley. 2011. *Designing for Situation Awareness: An Approach to User-Centered Design, Second Edition* (2nd ed.). CRC Press, Inc., Boca Raton, FL, USA.

[28] Davide Falessi, Shiva Nejati, Mehrdad Sabetzadeh, Lionel Briand, and Antonio Messina. 2011. SafeSlice: a model slicing and design safety inspection tool for SysML. In *Proc. of the 19th ACM SIGSOFT Symp. and the 13th European Conf. on Foundations of Software Engineering*. ACM, 460–463.

[29] Martin S. Feather and Lawrence Z. Markosian. 2013. Architecting and generalizing a safety case for critical condition detection software: an experience report. In *Proc. of the 1st Int'l WS on Assurance Cases for Software-Intensive Systems*. 29–33.

[30] K. Fellah and M. Guiatni. 2019. Tactile Display Design for Flight Envelope Protection and Situational Awareness. *IEEE Transactions on Haptics* 12, 1 (Jan 2019), 87–98.

[31] Qian Feng and Robyn R. Lutz. 2005. Bi-directional safety analysis of product lines. *Journal of Systems and Software* 78, 2 (2005), 111–127.

[32] Donald Firesmith. 2004. Engineering Safety Requirements, Safety Constraints, and Safety-Critical Requirements. *Journal of Object Technology* 3, 3 (2004), 27–42.

[33] Markus Funk. 2018. Human-drone interaction: Let's get ready for flying user interfaces! *Interactions* 25, 3 (2018), 78–81.

[34] Matthew C. Gombolay, Anna Bair, Cindy Huang, and Julie A. Shah. 2017. Computational design of mixed-initiative human-robot teaming that considers human factors: situational awareness, workload, and workflow preferences. *I. J. Robotics Res.* 36, 5-7 (2017), 597–617.

[35] Richard Hawkins, Ibrahim Habli, and Tim Kelly. 2013. Principled Construction of Software Safety Cases. In *Proc. of the Next Generation of System Assurance Approaches for Safety-Critical Systems WS of the 32nd Int'l Conf. on Computer Safety, Reliability and Security*.

[36] Andreas Holzinger. 2005. Usability engineering methods for software developers. *Commun. ACM* 48, 1 (2005), 71–74.

[37] Teton Valley News Julia Tellman. 2018. First-ever recorded drone-hot air balloon collision prompts safety conversation. https://www.postregister.com/news/local/first-ever-recorded-drone-hot-air-balloon-collision-prompts-safety/article_7cc41c24-6025-5aa6-b6dd-6d1ea5e85961.html. [Last accessed 01-06-2021].

[38] Shuanglong Kan. 2014. Traceability and model checking to support safety requirement verification. In *Proc. of the 22nd ACM SIGSOFT Int'l Symp. on Foundations of Software Engineering*. ACM, 783–786.

[39] Leah Kaufman and Brad Weed. 1998. Too much of a good thing?: Identifying and resolving bloat in the user interface. In *Proc. of the CHI 98 Conf. Summary on Human Factors in Computing Systems*. 207–208.

[40] Tim Kelly and Rob Weaver. 2004. The goal structuring notation–a safety argument notation. In *Proc. of the dependable systems and networks 2004 workshop on assurance cases*. Citeseer, 6.

[41] Tim P Kelly and John A McDermid. 2001. A systematic approach to safety case maintenance. *Reliability Engineering & System Safety* 71, 3 (2001), 271–284.

[42] L.T. Kohn, J.M. Corrigan, and M.s. Donaldson. 1999. To err is human, Building a safety health system. *Washington, DC: National Academy Press* (1999).

[43] Yasuhiro Kuriki and Toru Namerikawa. 2014. Consensus-based cooperative formation control with collision avoidance for a multi-UAV system. In *Proc. of the 2014 American Control Conf.* IEEE, 2077–2082.

[44] Nancy G. Leveson. 1995. *Safeware, System Safety and Computers*. Addison Wesley.

[45] Mai Skjott Linneberg and Steffen Korsgaard. 2019. Coding qualitative data: A synthesis guiding the novice. *Qualitative Research Journal* (2019).

[46] Yuanna Liu and Hao Lu. 2019. A strategy of multi-UAV cooperative path planning based on CCPSO. In *Proc. of the 2019 IEEE Int'l Conf. on Unmanned Systems*. IEEE, 328–333.

[47] Robyn R. Lutz and Robert M. Woodhouse. 1997. Requirements Analysis Using Forward and backward Search. *Ann. Software Eng.* 3 (1997), 459–475.

[48] Joanna McGrenere, Ronald M. Baecker, and Kellogg S. Booth. 2002. An Evaluation of a Multiple Interface Design Solution for Bloated Software. In *Proc. of the SIGCHI Conf. on Human Factors in Computing Systems* (Minneapolis, Minnesota, USA). ACM, 164–170.

[49] D.C. Nagel. 1998. Human error in aviation Operations. *Human factors in Aviation, E.L.Weiner and E.C.Nagel (Eds)* 19890047069, 34 (1998), 263–303.

[50] Saeid Nahavandi. 2017. Trusted autonomy between humans and robots: toward human-on-the-loop in robotics and autonomous systems. *IEEE Systems, Man, and Cybernetics Magazine* 3, 1 (2017), 10–17.

[51] Jakob Nielsen and Thomas K Landauer. 1993. A mathematical model of the finding of usability problems. In *Proc. of the INTERACT'93 and CHI'93 Conf. on Human Factors in Computing Systems*. 206–213.

[52] E Onal, C Craddock, and Mica Endsley. 2013. From Theory to Practice: How Designing for Situation Awareness Can Transform Confusing, Overloaded Shovel Operator Interfaces, Reduce Costs, and Increase Safety. In *Proc. of the Int'l Symp. on Automation and Robotics in Construction*.

[53] Parsif.al. 2021. Tool support for Systematic Literature Reviews. https://parsif.al. [Last accessed 01-06-2021].

[54] PX4. 2021. Open Source Flight Controller. https://px4.io. [Last accessed 01-06-2021].
[55] QGroundControl. 2021. Ground Control Station. http://qgroundcontrol.com. [Last accessed 01-06-2021].
[56] Nicolas Regis, Frédéric Dehais, Emmanuel Rachelson, Charles Thooris, Sergio Pizziol, Mickaël Causse, and Catherine Tessier. 2014. Formal Detection of Attentional Tunneling in Human Operator-Automation Interactions. *IEEE Transactions Human-Machine Systems* 44, 3 (2014), 326–336.
[57] Donald J. Reifer. 1979. Software Failure Modes and Effects Analysis. *IEEE Transactions on Reliability* R-28,3 (1979), 247–249.
[58] Mehrdad Sabetzadeh, Shiva Nejati, Lionel Briand, and Anne-Heidi Evensen Mills. 2011. Using SysML for modeling of safety-critical software-hardware interfaces: Guidelines and industry experience. In *Proc. of the IEEE 13th Int'l Symp. on High-Assurance Systems Engineering*. IEEE, 193–201.
[59] Neil R. Storey. 1996. *Safety Critical Computer Systems*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA.
[60] Tim Claudius Stratmann and Susanne Boll. 2016. Demon Hunt - The Role of Endsley's Demons of Situation Awareness in Maritime Accidents. In *Proc. of the Int'l Working Conf. on Human Error, Safety, and System Development*. Springer, 203–212.
[61] Kevin J. Sullivan, Joanne Bechta Dugan, and David Coppit. 1999. The Galileo Fault Tree Analysis Tool. In *Digest of Papers: FTCS-29, The Twenty-Ninth Annual Int'l Symp on Fault-Tolerant Computing*. IEEE Computer Society, 232–235.
[62] Hongyu Sun, Miriam Hauptman, and Robyn R. Lutz. 2007. Integrating Product-Line Fault Tree Analysis into AADL Models. In *Proc. of the 10th IEEE Int'l Symp. on High Assurance Systems Engineering*. IEEE Computer Society, 15–22.
[63] Dante Tezza and Marvin Andujar. 2019. The State-of-the-Art of Human–Drone Interaction: A Survey. *IEEE Access* 7 (2019), 167438–167454.
[64] Christopher D Wickens and Amy L Alexander. 2009. Attentional tunneling and task management in synthetic vision displays. *The International Journal of Aviation Psychology* 19, 2 (2009), 182–199.
[65] Xueyi Zou, Rob Alexander, and John McDermid. 2016. Testing method for multi-uav conflict resolution using agent-based simulation and multi-objective search. *Journal of Aerospace Information Systems* 13, 5 (2016), 191–203.