Human-Machine Interaction Fuzzing in Small Uncrewed Aerial Systems: From Simulation to Field

Theodore Chambers
University of Notre Dame
Dept. of Computer Science and Engineering
Notre Dame, IN, USA
tchambe2@nd.edu

Governors State Authors
Governors State University
Computer Science
University Park, IL, USA
emailhere

Other student authors
University of Notre Dame
Dept. of Computer Science and Engineering
Notre Dame, IN, USA
emails

Michael Vierhauser
University of Innsbrück
Dept name??
Innsbruck, Austria
michael.vierhauser@jku.at

Ankit Agrawal
Saint Louis University
Depart. of Computer Science
St. Louis, MO, USA
aagrawal@slu.edu

Jane Cleland-Huang
University of Notre Dame
Dept. of Computer Science and Engineering
Notre Dame, IN, USA
JaneHuang@nd.edu

ABSTRACT

This paper will probably have 10 authors by the time we finish, as several UGs and a visiting scholar will help with it. The rapid growth in deployment of small Uncrewed Aerial Systems (sUAS) has led to a rapid increase in flight-time incidents, many of which have involved human-related problems such as loss of situational awareness, process missteps, and insufficient safeguards. In turn, these problems point to inadequacies in system and user interface designs. Prior work in this area has shown how Fuzz Testing can be used to identify human-drone interaction errors in a simulation environment, and has proposed a progression of tests that theoretically lead to field deployments. However, the current process requires significant manual effort without automated support beyond the simulation stage. In this paper, we address this weakness by automating the testing pipeline using highly focused, multi-level fuzzing to discover the structure of associated fault-trees. We then generate test-cases with predicted levels of flakiness, and ultimately produce candidate safety-assurances cases including placeholders for evidence collected at various stages of the testing process. The end result is a streamlined process for detecting problems and designing and validating solutions that mitigate hazardous human-drone interaction failures. We perform an in-depth validation of our approach in a multi-vehicle system of autonomous sUAS, and discuss generalization of our findings into other robotics systems.

ACM Reference Format:
Theodore Chambers, Governors State Authors, Other student authors, Michael Vierhauser, Ankit Agrawal, and Jane Cleland-Huang. 2024. Human-Machine Interaction Fuzzing in Small Uncrewed Aerial Systems: From Simulation to Field . In *Proceedings of 47th International Conference on Software Engineering (ICSE 2025).* ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

ICSE 2025, April 2025, Ottowa, Canada

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

Prior studies have shown that 65% to 85% of reported accidents associated with Cyber-Physical Systems (CPS) have been attributed to human-related interaction 'errors' [2–5], with similar observations in the emergent domain of Small Uncrewed Aerial Systems (sUAS) [**?**]. For example, XXX report on a windy-day incident in which an autonomous sUAS reached its geofence, flew off its designated flight path, and ascended to an altitude of 734 feet above ground level (AGL) – far above the legal limit of 400 feet AGL. Their analysis of the incident indicated several human missteps when the remote pilot in charge (RPIC) failed to properly setup the geofence, had slight deviations in the expected RC Transmitter settings, and ultimately was unaware that the sUAS had ascended above legally allowed altitudes. Blaming the operator for making these types of mistakes is somewhat counterproductive. Instead, we need to design systems to be robust against normal human errors (REFS).

Chambers et al., [1] proposed a novel approach that leveraged fuzz-testing to detect potential human-machine interaction errors in sUAS applications. Their approach included a pipeline of three levels and two gateways based on a progression from simulation to real-world tests. At level 1 (L1) they applied fuzz-testing of potential human inputs in various flight modes and sUAS configurations. Outcomes were assessed by identifying deviations from the sUAS intended flight route. They then generated a profile of inputs, configurations, and flight deviations, and clustered the results to identifying groupings of failures which could be examined and ultimately mitigated. Their overall testing pipeline involved a second step with human-in-the-loop testing in simulation, and a manually executed safety gateway for assessing when the mitigated solution could be tested in the real-world. As such, their work focused primarily on identifying human-interaction failures, which would then be assessed and mitigated manually throughout the remainder of the pipeline.

The identified a number of problems. While Chambers et al., simply listed these problems, we categorize them into four groupings, each one emphasizing different facets of human-system interaction. They included *Configuration and Setup Errors* involve RPIC mistakes during flight preparation, such as incorrect configurations and RC Transmitter settings; *Interface Mismanagement* in which the RC Transmitter or the underlying source code provided unwanted and potentially dangerous interaction options and/or failed to respond reliably to human inputs; *Operational Decision Errors* reflecting poor judgement during flight operations, such as inappropriately timed mode changes or failure to operate the drone according to its current mode, both of which can severely affect operation safety and effectiveness; and finally *Situational Awareness* problems in which the RPIC failed to understand what the autonomous system was actually doing. Each category pinpoints an area in which improvements in training, system design, and operational protocols are needed to enhance safety and efficiency in sUAS operations.

In this paper, we build upon the work by Chambers et al., focusing on downstream aspects of their testing pipeline that were largely unaddressed in their work. Whereas this prior work focused on detecting design problems, our new work focuses squarely on the testing pipeline. Our starting point is a grouping of hazardous outcomes found in the Level L1 fuzz testing environment. Building upon our previous example, one grouping of errors that was identified in the prior work was based on the combination of incorrect geofence settings, throttle settings, and wind. However, without understanding the interplay between these three factors, it is difficult to properly and reliably mitigate the solution, and provide assurance that the problem is adequately addressed. The work in this paper focuses on this part of the process. Given a category of problems, we perform a second round of highly focused fuzz tests in order to generate an associated fault-tree. We generate associated tests cases and leverage results from the fuzz tests to determine the *flakiness* and the *sensitivity* of each test. We further generate a *candidate safety assurance case* and systematically populate it with design decisions and appropriate test results in order to build a case that the human-interaction error has been properly mitigated.

Our work makes three key research contributions. First, it vastly improves the prior approach for detecting human-machine interaction problems in G1 in order to fully automate the process and streamline the pipeline. Second, it improves the fidelity of the human-proxy to introduce slight timing delays as expected with real humans in the field. Third, it introduces a new stage in the testing pipeline, with a focus on generating tests and preliminary interactive safety-cases to aid the transition from simulation to physical field-tests. The increased levels of automation require several additional innovations including a new domain-specific language for specifying UAV behaviors, integrated into a feature model that supports the automated generation of test cases.

More coming..

The remainder of the paper is laid out as follows: blah blah

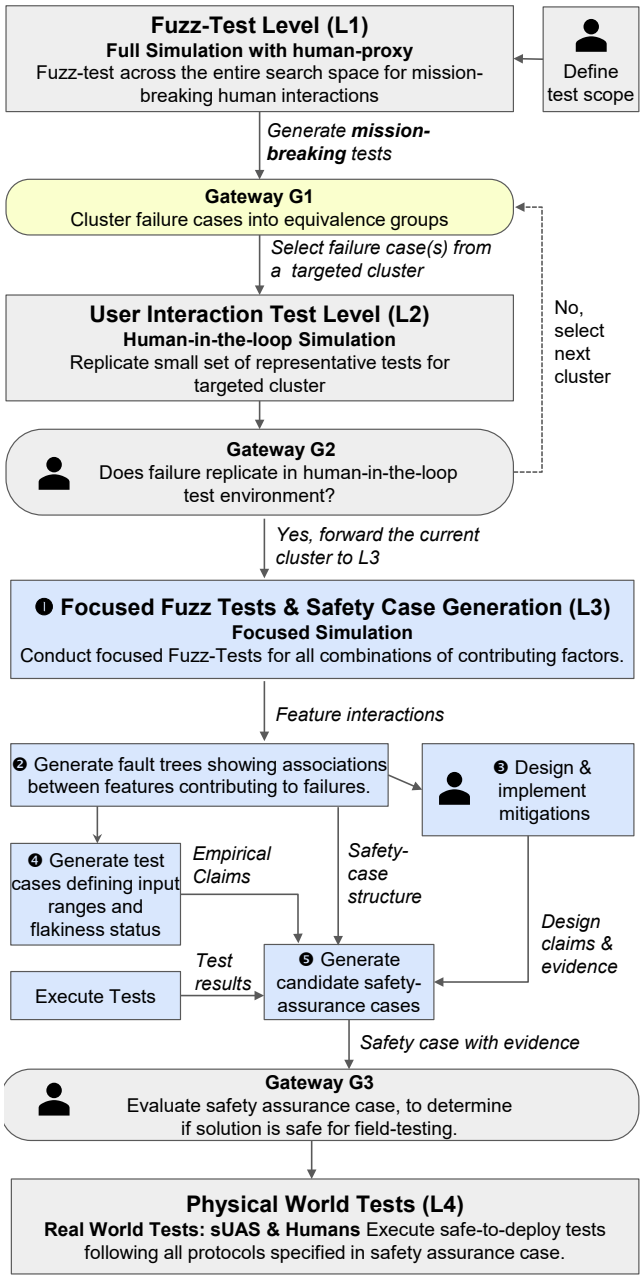

Figure 1: The Fuzz-Testing pipeline for identifying and validating mitigations for Human-Drone Interaction failures. Gray elements are taken from prior work by Chambers et al., [1]. Blue nodes represent novel contributions of this paper, while the yellow node represents an augmented component.

2 OVERVIEW OF THE PROCESS

In this section we briefly summarize the process described by Chambers et al., [1] and differentiate our contributions. We then take a deeper dive into the proposed process for automating and guiding aspects of the testing and safety-assurance pipeline, drawing upon an illustrated example based upon the previously discussed failure.

2.1 The Baseline HiFUZZ Approach

The original HiFUZZ process included three levels: L1, L2, and L3, separated by two gateways. A short description of each step

- *Level L1:* Level L1 focuses on

2.2 The Augmented Testing Pipeline

Here we provide an overview of the changes. We won't give details here because I think we need a separate section for the clustering and DOWN SELECTION and a separate section for the automation. Our augmented testing process interjects a new level L3 (shown in blue in Figure 1), representing the core contribution of this paper. In addition, we have made significant modifications to Gateway G1, to significantly increase the automation of our overall pipeline. will expand later.

3 IMPROVED HMI FAILURE DETECTION

This section is going to describe the improved G1 gateway. Basically, it automates what used to be a very manual approach.

@jchTODO: Theo should generate at least five different sets of test data for experimentation, then the summer student working on this can use that data without needing to run the entire fuzzer.

4 FROM SIMULATION TO FIELD-TESTS

In this section we focus on the automation of the testing process to facilitate the transition from simulation to field tests. We illustrate each step of the process using our running example from the introduction. The input to this level (L3) is a single clustering from Gateway G1. As in our example failure cluster, this included *wind*, *incorrect set geofence action*, and *a slightly raised throttle position*. Our goal in this stage of the pipeline is to start with a cluster (using this one as an example) and to automatically generate test cases and a preliminary safety case for evaluation. We achieve this through a series of steps, labeled 1-5. In step 1, we utilize a focused form of fuzzing to probe the relationships between elements in the cluster, and as a result we generate a fault tree (step 2). We then use the fault tree to guide the manual process of designing and implementing appropriate mitigations (step 3) and to generate automated tests (step 4) for near-term execution in simulation and future execution in real-world environments. Finally, we generate a preliminary safety assurance case (step 5) which aggregates fault-related data with claims and evidence collected from the previous steps. All aspects of this level are automated except for Step 3.

4.1 Fuzzing Probes

Unlike prior work which accepted a clustering of features as the final grouping, we use them as the starting point for deeper analysis. Let's assume for now, that the G1 gateway identifies a cluster characterized by three elements A, B, and C. Our goal is to determine the feature interactions of these elements. In other words, we want to model the *fault tree structure* of the elements to determine if the failure occurs when we have $A \wedge B \wedge C$, $A \vee B \vee C$, or some other combination such as $A \wedge (B \vee C)$, or even the presence of an XOR relationship such as $A \wedge (B \oplus C)$. Learning the structure of the failure-inducing elements is essential for downstream testing and safety assurance.

We therefore generate an iterative *Fuzzing Probe* to unearth these associations. With only three elements, it is possible to fuzz each combination i.e., A, B, C, AB, AC, BC, and ABC. We start by defining the fuzz space for each of the three elements. For a given element (e.g., A), this could be a discrete set of states (e.g., $A = \{a_1, a_2, \ldots, a_n\}$), or it could be a continuous range of values, such as $A = [a_{min}, a_{max}]$ where a_{min} and a_max denote the minimum and maximum values of the range. By defining A, B, and C, we define the scope of the fuzz space. Our interest in this work is to explore human-machine interaction failures, and therefore we assume that at least one element has a human factor.

For each feasible combination of A, B, and C, as depicted in the three left hand columns of Table 1, we execute a *fuzz probe*, meaning that we perform fuzz testing across the entire space. To limit the scope of the probe, we first establish its static context, representing variables that the L1 level tests and G1 Gateway analysis showed as 'don't care' conditions for the cluster.

Assuming (for now) only three or four error factors characterize each failure cluster, we start by fuzzing each candidate combination, searching for evidence of a failure. For each combination of elements we run N fuzz tests (where N is determined according to the potential size of the combinatorial fuzzing space. This is discussed in Section X. Need more (see comment).) Following the first round of tests, we mark the combination is TRUE (i.e., producing a HMIF (human machine interaction failure), iif at least one error outcome is observed. The following table shows three potential observations where A, B, and C represent some set of elects (e.g., Wind, Throttle, and Geofence actions). At some point we might change this table to show PROBABILITY OF FAILURE instead of just True or False! Alternatively introduce that later.

Elements			Example Observations		
A	**B**	**C**	Case 1	Case 2	Case 3
T	T	T	●	●	
T	T	F	●	●	●
T	F	T		●	●
T	F	F		●	
F	T	T		●	
F	T	F		●	
F	F	T		●	
F	F	F			
Outcomes			**A ∧ B**	**A ∨ B ∨ C**	**A ∧ (B ⊕ C)**

Table 1: Truth table for logical expressions

In the first example (Case 1) there are two combinations for which failures were observed. These are ABC, and AB$\bar{C}$. Since the outcome is true for any value of C, we can simplify this to AB (i.e., A $\wedge B$). The second case reduces to $A \vee B \vee C$, and the second represents $A \wedge (B \oplus C)$ meaning it it occurs when the state A is present with B XOR C.

So, the minimal logic formula for ABCABC when CC is a "don't care" is ABAB.

Our probe performs the following steps Convert to algorithm later.

(1) Fuzz each individual factor separately to determine if it

REFERENCES

[1] Theodore Chambers, Michael Vierhauser, Ankit Agrawal, Michael Murphy, Jason Matthew Brauer, Salil Purandare, Myra B. Cohen, and Jane Cleland-Huang. 2024. HIFuzz: Human Interaction Fuzzing for Small Unmanned Aerial Vehicles. In *Proceedings of the CHI Conference on Human Factors in Computing Systems*. ACM, Honolulu, HI, USA, 1–14. https://doi.org/10.1145/3613904.3642958

[2] Chin-Feng Fan, Ching-Chieh Chan, Hsiang-Yu Yu, and Swu Yih. 2018. A simulation platform for human-machine interaction safety analysis of cyber-physical systems. *International journal of industrial ergonomics* 68 (2018), 89–100.

[3] 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).

[4] 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. https://doi.org/10.1109/2.910904

[5] 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 Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*. 8–19.