FROM THE CONCEPT OF OPERATIONS TOWARDS AN OPERATIONAL DESIGN DOMAIN FOR SAFE AI IN AVIATION

Thomas Stefani*, Akshay Anilkumar Girija*, Ryan Mut*, Sven Hallerbach*, Thomas Krüger*
* Institute for AI-Security, German Aerospace Center, Ulm, Germany

Abstract

In the project RESILIENZ at the German Aerospace Center, a systematic approach for combining operational scenarios from Concept of Operations (ConOps) to capture specific operational ranges and limitations, is investigated. Based on an exemplary aviation use case - collision avoidance -, the corresponding ConOps is defined and used as a basis to derive the Operational Design Domain (ODD). Since ODD is one of the pillars for developing safe AI-based systems this enables meeting high-level system requirements and in the future resilience for failure tolerant AI systems. In this paper, a model-based system engineering framework is introduced in which scenarios are generated from the ConOps description in a highly automated way. In total 50 scenarios were generated and evaluated. This dataset of the operational scenario can be used for testing the initial ODD coverage. The framework in this paper ensures a systematic and highly automated approach from describing the ConOps towards deriving and testing an ODD in the aviation domain.

Keywords

AI Systems Engineering, Operational Design Domain, Scenario Generation, Scenario-based Testing

1. INTRODUCTION

In recent years, the research on applied Artificial Intelligence (AI) in various aspects of aviation has advanced exponentially [1]. Consequently, the European Union Aviation Safety Agency (EASA) published the first concept paper of their AI roadmap in April 2021, demonstrating a possible way towards certification for AI applications in aviation [2].
Within this guidance, the ODD plays a crucial role in increasing the safety of AI-based systems, especially in the context of increasing autonomy of the system. ODD refers to the specific conditions and constraints under which a system containing AI components is intended to operate safely. Defining and adhering to a well-defined ODD is essential for mitigating risks and ensuring the safe deployment of AI-based systems [3]. Initially ODD was described in the automotive sector based on the document SAE J3016 [4] for autonomous driving [5] [6].
The process of adapting the approach from the automotive sector to the aviation domain is a remaining research topic [7]. As mentioned by EASA, the ConOps play an important role in setting the basis for an ODD description by providing detailed descriptions of operational scenarios [2]. In this paper an exemplary operational scenario of *collision avoidance* is chosen. Based on this scenario a potential process of describing the corresponding ConOps and deriving the ODD is shown. However, to verify your system and therefore the ODD compliance, scenario-based testing is a suitable approach. Due to reasons of efficiency to a large extend this testing is done in a virtual environment. Virtual testing is crucial for gathering the relevant data for assessing the performance and safety of an AI-based system. For this the open source flight simulators FlightGear [8] is used [9].
A scenario generator is created using a python script which generates slight variations of an midair collision scenario with changing environmental conditions such as "time of the day". Further the logging and storing of the data is automized, allowing efficient evaluation of the gathered results. As a result a framework was developed which demonstrates the process from ConOps to ODD description with included scenario-based testing of the operational scenarios. Throughout the engineering process Model-Based System Engineering (MBSE) is used in order to handle engineering complexity [10] [11].

2. RELATED WORKS

2.1. Concept of Operations

In general, a ConOps is a concise document that outlines the high-level description of how a system, project, or operation is intended to function and achieve its objectives [12]. In the context of this paper ConOps provides a clear understanding of how aircrafts are intended to operate, the range of conditions they can handle, and the constraints that govern their behavior. By formulating a comprehensive ConOps, developers and stakeholders can establish a common understanding of the system's capabilities and limitations, which are fundamental for defining the ODD. Figure 1 represents an operational procedure (runway, climb, cruise, avoid collision and descent) of an aircraft [7] [12].

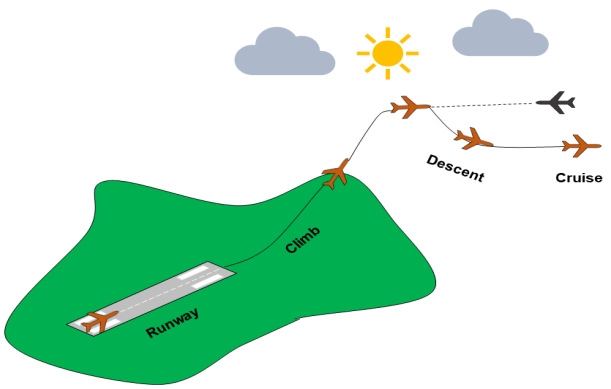

FIG 1. Schematic representation of an aircraft's ConOps [7]

In this paper, one operational scenario (collision avoidance) of the ConOps is described. This operational scenario is centered around avoiding collision using an AI-based system [13] [14] which utilizes Automatic Dependent Surveillance-Broadcast (ADS-B) data. This device receives and transmits signals, allowing the aircraft to broadcast their position and other information for air traffic surveillance in order to avoid collision. The AI system analyses this data to predict an advisory and to provide guidance to the pilot.
Within the ConOps, stakeholders collaborate to identify key components, such as environmental factors and operational constraints. Environmental factors, including weather conditions, geographical settings, are crucial in determining the extent to which an aircraft can safely and effectively operate. For instance, an autonomous aircraft could operate at its best during favorable weather conditions with good visibility. However, it might be required to adjust its flight speed during special weather conditions such as rain or snow. Similarly, operational limitations such as airspeed restrictions and airspace congestion play a role in establishing the boundaries of the ODD.

2.2. Operational Design Domain

ODD is an important concept in the context of autonomous vehicles, including self-driving cars, drones, and unmanned aerial vehicles. It refers to the specific operating conditions and constraints under which an autonomous vehicle can safely and effectively operate. The ODD defines the boundaries within which the autonomous system's capabilities and limitations are valid, ensuring the vehicle function as intended and avoids unsafe situations. According to the Society of Automotive Engineers (SAE) J3016 standard, ODD is defined as "the specific conditions under which a given driving automation system or feature thereof is designed to function, including, but not limited to, environmental, geographical, and time-of-day restrictions, and the requisite presence or absence of certain traffic or roadway characteristics [4]."
The importance of ODD lies in several aspects [6]:

1) Safety: Defining a well-defined ODD helps ensure that the autonomous system operates safely and reliably within its designed limits. By identifying conditions in which the system may not perform optimally or may encounter challenges, it becomes easier to implement appropriate safety measures and fallback strategies.
2) Risk Mitigation: ODD allows developers and regulators to focus on specific scenarios and assess potential risks associated with autonomous operation. By understanding the ODD's boundaries, engineers can concentrate on refining the system's capabilities for the most common and critical driving conditions.
3) Transparent Communication: Defining the ODD is crucial for effectively communicating the autonomous system's capabilities to users, operators, and other stakeholders. This transparency helps manage expectations and avoids situations where users assume the system can handle tasks beyond its intended design.

2.3. ConOps for AI Application

In EASAs Concept Paper several building blocks (e.g. AI assurance, Human Factors for AI, AI Safety Risk Mitigation) for trustworthy AI are described. Therein the role of ConOps for the AI application is mentioned since the data-driven nature of AI requires precise operational scenario definitions and constraints to ensure sufficient and representative data is collected for the training data sets. Therefore an objective of the concept paper requires the applicant to define and document the ConOps for the AI-based system. Special focus should be put on the definition of the ODD and the capturing of specific operational limitations and assumptions.
In Figure 2 the interrelationship between ConOps and the Operational Domain (OD) from the concept paper is shown. Here a distinction between the SAEs [4] definition of ODD and EASAs definition has to be mentioned. SAEs ODD corresponds to EASAs operational domain. EASA only defines an ODD at the AI constituent level [2]. In this paper the definition from SAE is used where ODD describes the domain of for the whole system, thus aircraft.
The ConOps contains several operational scenarios. The system in these operating scenarios runs on specific operating parameters. However the operational domain potentially limits the ranges for certain parameters. The exact boundary line is of great importance, since it separates the "safe" and "unsafe" zone. Consequently it is mentioned in the concept paper [2] that OD (in our case ODD) is continuously refined during the learning process. As a consequence, it is important to note that before being able of iterating through an ODD definition, first it is crucial to have an initial ODD from where refinement can take place.

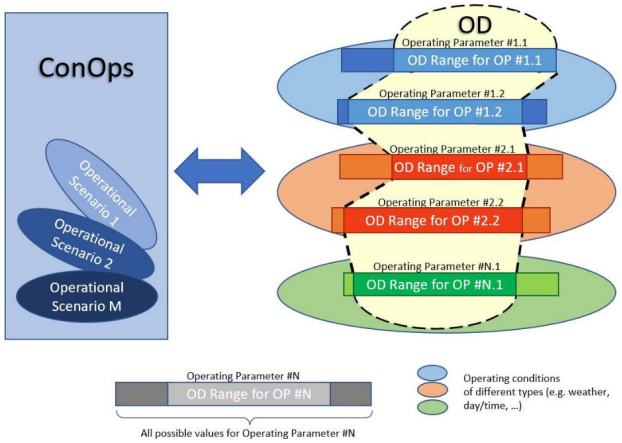

FIG 2. Interrelationship between ConOps and ODD [2]

2.4. Scenario-based Testing

Ensuring the safe behavior of complex systems across a range of relevant scenarios is a challenging task. Due to the high amount of test samples, relying solely on real-world testing for the verification and validation of highly automated systems is infeasible. To complement these efforts, the utilization of scenario-based testing can be applied. This methodology involves testing the highly automated system in different scenarios [15] [16].
In aviation operational scenarios are for example "collision avoidance" or "landing". However one operational scenario such as collision avoidance consists of various variations for this particular instance. These various scenarios consists of operating parameters that describe its various aspects, such as the initial position, speed and direction of the system-under-test. During the generation of test cases for these scenarios, concrete values are assigned to all parameters of the considered scenario [17]. Recognizing the impracticality of testing the system across in all possible scenario instances, it becomes necessary to select *relevant* test cases that challenge the system under test. A test case yields valuable information when it reveals potential faulty behavior in the system. This paper focuses on the scenario-

based data generation inline with EASAs concept paper [2], which in a subsequent step can be used to validate your system [18] [19].

3. CHALLENGES IN THE ENGINEERING PROCESS

After considering the ConOps for AI application in 2.3 and the importance of correctly defining and testing the ODD boundary, the question arises on what the next engineering steps are and how they are conducted. For this, Figure 3 is adjusted based on Figure 2. The process starts with an operational scenario description. As an example of an operational scenario in the aviation domain collision avoidance (CA) is considered. Splitting an operational scenario description provides a logical and a concrete scenario [20]. The logical scenario describes the sequence of scenes or actions. For the safety validation of these systems, it is important to correctly define the ODD boundary. From the logical scenario description, the system requirements and other environmental requirements can be extracted. Based on these requirements, concrete scenarios can be defined where all the attributes in the logical scenarios are subjected to precise quantification [21].

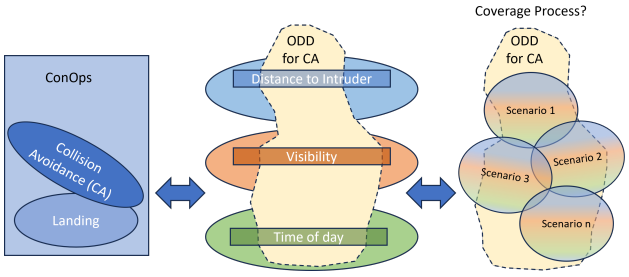

FIG 3. Challenges in the engineering process for demonstrating ODD coverage

Figure 3 also highlights the wide range of attributes and parameters that can be used for generating concrete scenarios. These attributes and parameter contribute to initial ODD description for an system, particularly in the context of a critical use case such as collision avoidance, which demands execution within a well-defined environment. There are some attributes within the concrete scenarios which also has continuous parameters such as visibility.
Now, the question arises how to generate scenarios in an methodical approach in order to achieve full ODD coverage [21]. A consequent challenge is to generate correct and relevant scenarios based on continuous parameters to test the system. Even though the test method is mostly performed using a simulation environment, the evaluation of these scenarios is critical and depend on correct metrics [21].

4. USE CASE - COLLISION AVOIDANCE

As mentioned in chapter 3 there are remaining challenges in the novel engineering process when AI constituents are involved in the safety critical aviation domain. In order to showcase a potential concept to investigate this issue, the exemplary use case/operational scenario of collision avoidance is chosen. An aircraft equipped with an AI-assistance functions for mid air collision avoidance [13] is therefore tested in a methodical approach. The concept orients on EASAs AI objectives for safety and security. The aim is to demonstrate the process from creating the relevant part of the ConOps and deriving an initial set for ODD definition. Later in the project testing all operating parameters during operation will be needed for the verification and validation of the system. However this requires a methodical approach especially towards the edge cases, where operating safety is crucial.

4.1. AI-based Collision Avoidance System

For the collision avoidance use case the ACAS sXu [13] is chosen as an AI-based system. The ACAS sXu is a specialized version of ACAS Xu developed under the supervision of the Federal Aviation Administration, with a primary focus on unmanned aircraft collision avoidance. The system utilizes numeric lookup tables, which are relatively large in size, to facilitate decision-making processes. However, one of the significant challenges encountered during the development was limited memory availability, necessitating the need for extensive table compression. In response, the developers adopted a DNN approximation technique, reducing the table size by a factor of 1000. Despite this advancement, the central concern remains how to verify the reliability and safety of DNN-based systems. The ACAS sXu faced significant challenges due to the limited memory resources available for onboard unmanned aircraft systems. The numeric lookup tables, being large in size, posed a storage predicament. To address this, the developers employed deep neural network approximation to drastically shrink the table sizes, ensuring efficient memory utilization. However, a crucial question arose regarding the trustworthiness of DNNs for critical decision-making tasks in collision avoidance systems. Robustness and reliability are essential for aviation safety, raising concerns about the local robustness of deep neural networks across various flight conditions [13] [14].

4.2. ConOps for the Use Case

Since the collision avoidance use case is taken as one example from the many possible operational scenarios, the ConOps description in this paper focuses only on CA. This operational scenarios contains the following items:

- Own Aircraft
- Intruder
- AI system for collision avoidance (ACAS sXu)
- ADS-B system
- Environmental conditions (e.g. weather)

In this specific operational scenario, an own aircraft is flying at 20.000 feet altitude with provided latitude and longitude coordinates. Simultaneously, an intruder aircraft approaches from a randomized direction at the same altitude, with a 20 km separation. By using the real-time ADS-B data, the AI system predicts an advisory for collision avoidance. According to the advisory of the system the pilots follows instructions and avoids collision. This scenario runs in varying weather conditions. Based on these factors a first version of the ODD is developed, incorporating parameters such as speed, latitude, longitude range, etc. which outlines the permissible conditions for the effective operation of the collision avoidance system. The sequential steps of the operational scenario are as follows:

1) Own Aircraft flying at an altitude of 20.000 feet with a speed of 400-500 knots.
2) The autopilot is activated keeping the own aircraft on course.
3) Weather is clear/rainy/windy/cloudy.
4) An intruder aircraft with randomized position and path is potentially approaching the own aircraft.
5) If intruder enters collision volume ADS-B system is activated.

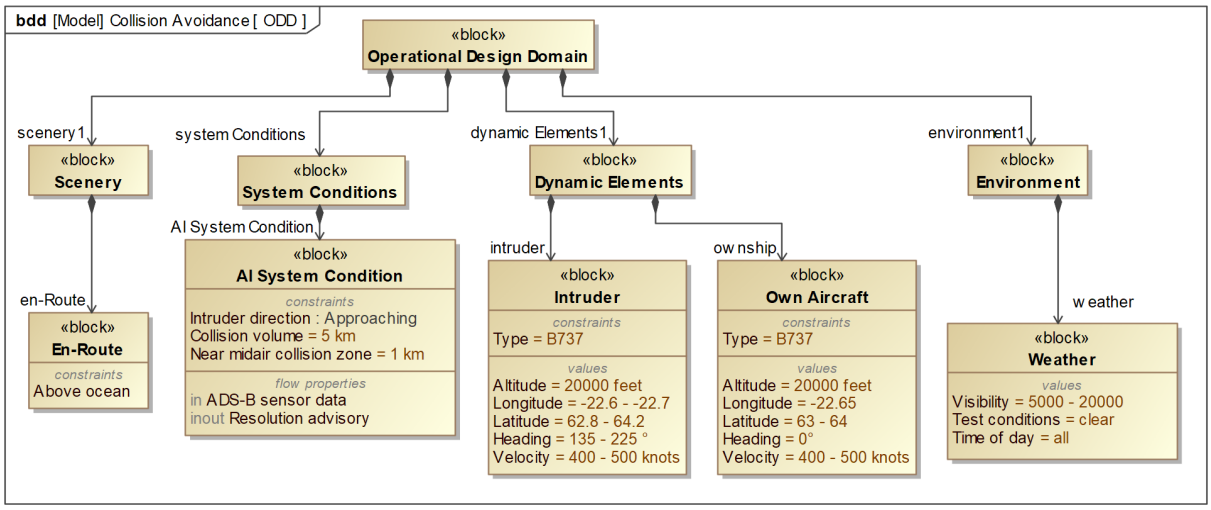

FIG 4. Simplified ODD description for collision avoidance use case based on operational scenarios description

6) The AI system processes ADS-B data, analyzing the position and velocity information.
7) Based on the input data, the AI system generates an advisory prediction.
8) The pilot receives the input from the AI system and takes appropriate action to prevent potential collision.

Based on the operational scenario description, operational requirements are:

1) The AI system should be activated within 5 km radius to generate a timely and effective resolution advisory.
2) The system should able to assure whether the intruder aircraft is moving towards own aircraft or not
3) The AI system must process ADS-B sensor data and generate an advisory for collision avoidance.
4) The ADS-B sensor data should include real-time velocity updates for intruder aircraft.

4.3. ODD for the Use Case

Based on the ConOps description in 4.2 an initial set of ODD blocks and parameters is created. For this representation a MBSE software based on SysML was used. The block definition diagram in Figure 4 shows the ODD consisting of four blocks, namely *Scenery, System Conditions, Dynamic Elements* and *Environment*. For the present use case, scenery depicts an en-route situation above the ocean. In these scenarios, two dynamic elements are present namely intruder and own aircraft. Each elements has its own attributes such as altitude, longitude, latitude, velocity and heading angle. Each attribute has its own specific value ranges. The environment is limited to weather conditions such as visibility and time of day ranging from day to night. The value for visibility describes the visible range in meters. If no value is provided the weather is *clear*. Additional to the ODD description from ASAM [22] a separate block for system conditions is used. Based on the current ODD description a critical prerequisite is the ADS-B system's capability to detect intruders within a 5 km radius, thereby enabling the activation of AI system. Beyond a 5 km radius the system lies outside of the ODD. A layer of criticality for the scenarios is defined by the near midair collision zone (NMACZ) of 1 km. It shall not be fallen below this separation level. From the data relevance point of view this determines an initial edge case.

4.4. FlightGear Simulator

FlightGear [8] is a free and open-source flight simulator software. It provides a realistic virtual environment for simulating various types of aircraft, in a wide range of scenarios and weather conditions.
FlightGear is built on the property tree which is a data structure used to store and manage various simulation-related information. It acts as a central repository for storing the state and properties of the virtual environment such as aircraft, systems, and other elements within the simulation. The property tree essentially organizes data in a hierarchical manner, where each property is identified by a unique path and can hold values such as numbers, strings and vectors. The property tree was used to exchange and update relevant information of the operational scenarios in real-time. For example, the property tree might store data related to aircraft position, engine parameters, weather conditions, control surfaces, and more. Information such as position, heading, speed, latitude, longitude etc. was used to simulate the ADS-B data.

4.5. Environment

The importance of an operational environment description in scenario-based testing is important as it serves as the foundation for creating realistic scenarios that closely mimic real world conditions. Testing scenarios above the ocean domain is simplified due to the absence of complex terrestrial obstacles, while the main focus is to detect the intruder rather than avoiding complex terrain structures. For this purpose the North Atlantic region is chosen. Figure 5 depicts the flying environment, with the lower half showing FlightGear's user interface displaying the map of the flight path.

5. FRAMEWORK

The aim of the framework presented in this paper is to address the challenges shown in chapter 3 and demonstrate a potential methodical approach towards using the ConOps description for ODD creation in the aviation domain. Further, the goal is to showcase how ODD definition can be used for the automated generation of *relevant* scenarios, which later can be used for the ODD coverage process.
Figure 6 shows the framework in a MBSE activity diagram. The color code is highlighting which steps are manual, semi-automated and automated. The method starts with the ex-

FIG 5. En-Route environment for the collision avoidance

emplary ConOps description of the operational scenario for collision avoidance in chapter 4.2. This serves as a basis to manually derive the operating parameters for the ODD as shown in Figure 4. The ODD description is implemented in MBSE tool from where a machine readable csv table can be exported. Using python script, a scenario file was generated which takes the ODD operating parameters and generates two outputs:

- a FlightGear readable flightplan for the intruder in xml
- a fgfs-source txt file containing the settings information

These two files contain all information for a specific scenario. Slight variation and randomization is implemented in the code for the scenario generation. Each new scenario gets a new ID and repository. The scenarios can be initiated and executed automatically in FlightGear. During runtime all operating parameters of the ODD and video data is stored with the same scenario ID. In a next step the results are evaluated. Especially in the beginning of this testing framework the need for adjustment arises. Therefore the scenario python script can be adapted iteratively.

5.1. Test Objectives

In the context of the use case, the main goal was to generate *relevant* scenario-based data for the ODD coverage process. Relevant data here refers to two requirements: Firstly, the intruder is entering the collision volume of 5km and secondly, the intruder approaching towards the own aircraft. Critical scenarios are identified when midair separation is below 1 km. In general the tests objectives are as follows:

- Functionality: Testing the general functionality of the framework
- Edge cases: Generate relevant data on the exemplary collision avoidance use case for potentially demonstrating ODD coverage
- Environmental conditions: Generate data on the use case for a wide range of operating conditions (e.g. weather)

5.2. Scenario Generation

Each scenario includes items that are listed in the ODD representation as shown in Figure 4. Both the own aircraft and the intruder are given deterministic behavior with slight variations for each scenario. For example the intruder is spawned in a conical area of around 20km in front of the intruder. Therefore the intruder is approaching the own aircraft from either left, center, right or moving away. The trajectory of the intruder is slightly randomized in each scenario by creating three waypoints in the xml flightplan. Scenarios also distinguish based on the applied environmental conditions like weather and time of the day.

5.3. Scenario Execution

The flightplan file is stored in $FG ROOT. When FlighGear is being called from a command line together with the fgfs-source file, a specific scenario is loaded which accesses the flightplan for this specific scenario. Parallel an ODD monitoring python script is started, which accesses the property tree in FlightGear and logs the required values of the ODD.

5.4. Data Storage

The test cases and corresponding results are saved as single log files. It consists of a series of test case description and ODD requirement which need to be tested. The logged data consists of the following:

- Generated during scenario generation: Scenario description and test conditions such as the position of the intruder that has been summoned, weather conditions, waypoints, velocities, heading angles etc.
- Generated during FlighGear runtime: All values of the dynamic elements that are listed in the ODD description (e.g. altitude, latitude, longitude, etc.)
- Generated during FlightGear runtime: Required values that are not directly available in the property tree and therefore need to be calculated (e.g. intruder direction: approaching/moving away, distance to intruder, etc.) the distance between intruder and own aircraft with corresponding timestamps in a .json file
- Generated during FlighGear runtime: Recorded video data with the corresponding ID of the test case

6. RESULTS

The proposed framework was established and tested using an exemplary operational scenario based on collision avoidance from ConOps. Based on this example, a simplified version of the ODD was created and implemented in an MBSE software. This process was executed manually and in a simplified manner with a reduced number of operating parameters, since trying to capture the reality as close as possible would lead to high demand of labour. Especially when operating parameters are interlinked the complexity of the system increases strongly. However, the goal was to demonstrate general feasibility of generating relevant scenario-based data, that can be used for either refine the initially developed ODD, or demonstrate ODD coverage. In total 50 scenarios were created, stored and categorized in a short period of time. Throughout the process, MBSE was utilized to enhance the design, manage complexity and facilitate collaboration within the developer team.

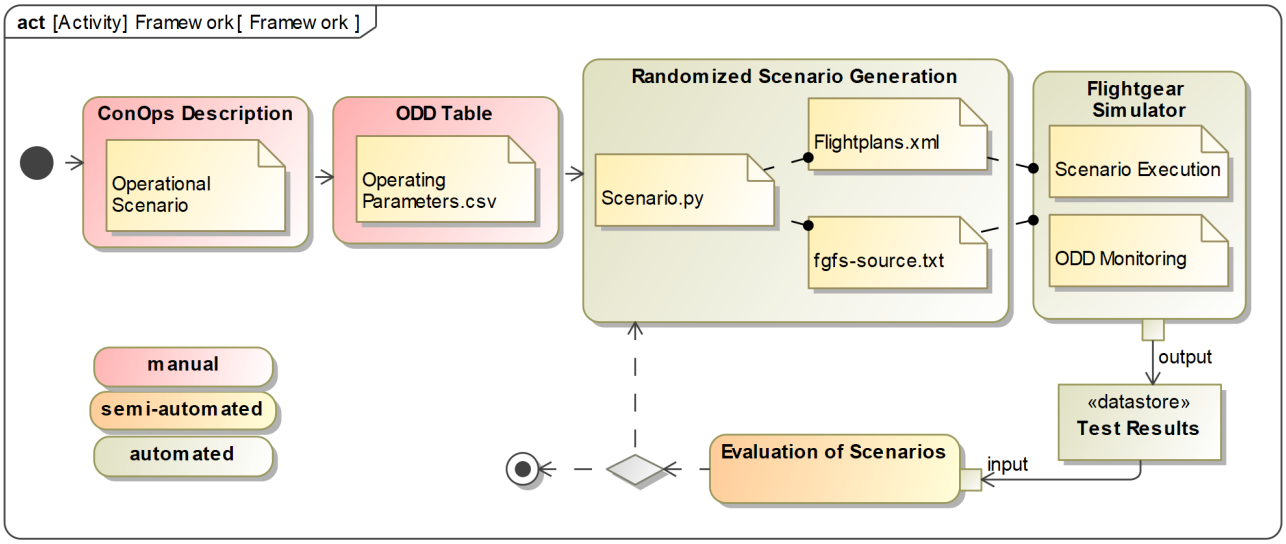

FIG 6. Overview of the framework: From ConOps & ODD to scenario generation, testing and evaluation.

6.1. Scenario Generator

The scenarios were generated using a python script. The own aircraft's trajectory follows a linear path along the longitude, while the latitude varies between 63°-64°. Altitude, heading and velocity of the own aircraft remain constant throughout the scenario. In a conical area in front of the own aircraft the intruder appears with an approx. 20 km separation. The heading angle of the intruder can vary between 135°-225° but with an adapting probability, in order to have a higher chance of getting a collision scenario. The altitude remains constant for both aircraft and the collision scenario can change between head-on or side-on collision scenarios. Further a 5 km radius is chosen to initially detect the presence of the Intruder aircraft. This exemplary chosen value represents the operational parameter in which the AI-system should start operating. As long as the intruder is within the 5 km radius, the collision avoidance system is expected to remain active. An additional circle with a 1 km radius centered around the own aircraft marks a critical area. Detection of the Intruder aircraft within this smaller circle indicates a *critical* scenario. If detection occurs within the 5 km radius but outside the 1 km radius circle, the scenario is defined *relevant*. These initial ODD values were chosen as an example to showcase the process of the framework.

6.2. Scenario and Evaluation

The assessment of preconditions for the collision avoidance involves a diverse range of temporal conditions. The evaluation starts with the initialization of the own aircraft at a specified origin point. The generation of scenarios is designed to emphasize the prevalence of relevant scenarios relative to non-relevant ones. As illustrated in Figure 7, the graph shows the creation of relevant scenarios, encompassing intruder approaches from the left, right, or head-on, occurring under varying time-of-day conditions. The operational scenario categorizes an intruder's movement as 'moving away' if it fails to converge with the own aircraft; conversely, it designates 'approaching' when the aircraft separation reduces. Of the randomly generated 50 test case scenarios, 46 produce scenarios where the intruder is converging towards the own aircraft. The calculation of inter-aircraft distance is based on corresponding longitude and latitude. Determining the intruders placement, whether it is summoned on the left, right, or ahead, is also done through longitude and latitude calculations. Diagram 7 further highlights that among the 46 scenarios, 14 involve intruders approaching from the left, 16 from the right, and 16 head-on.

The requirement to activate the AI system is an aircraft separation below 5km. In total, 42 out of the 46 scenarios satisfy this condition. The subsequent task for the AI system encompasses predicting the intruder's trajectory. If the trajectories intersect, the system provides collision-avoidance advisories. Therefore, an additional requirement is defined, mandating intruder detection within a 1-kilometer range. Notably, the 1-kilometer range primarily serves to verify the intruder proximity to the own aircraft, thereby generating critical scenarios wherein the AI system must deliver collision-avoidance advisories. As an ODD requirement, the own aircraft must possess the capability to anticipate the intruder's trajectory and, when necessary, proactively issue collision-avoidance advisories. The response time of the AI system is equally pivotal in this context, leading to 21 scenarios that satisfy detection within a 1-kilometer radius, influenced by the randomization function in the Python script. The visibility of the aircraft has been randomly varied within a range of 5 to 20 kilometers. Furthermore, scenarios involving clear visibility are also included. Likewise, time-of-day conditions, as depicted in Figure 7, have been selected to encompass periods from dusk till dawn, yet without affecting the detection of intruder aircraft. It is noteworthy that visibility conditions and time of day will bear significance in the context of computer vision-based AI systems.

As mentioned before, 50 test cases for collision avoidance were generated in a highly automated way containing all operating parameters described in the ODD table in section 4.3. Considering the initial ODD definition the framework provides a potential approach for generating sufficient data in a methodical way. By implementing an overlap on the operating parameters and generating a high amount of scenarios, ODD coverage can potentially be shown. The evaluation process needs further automation in case of higher number of operating parameters. Then it is necessary to plot which ranges of the parameters are represented within the data. Towards the ODD boundary a search space can be implemented filtering the critical scenarios for testing purposes.

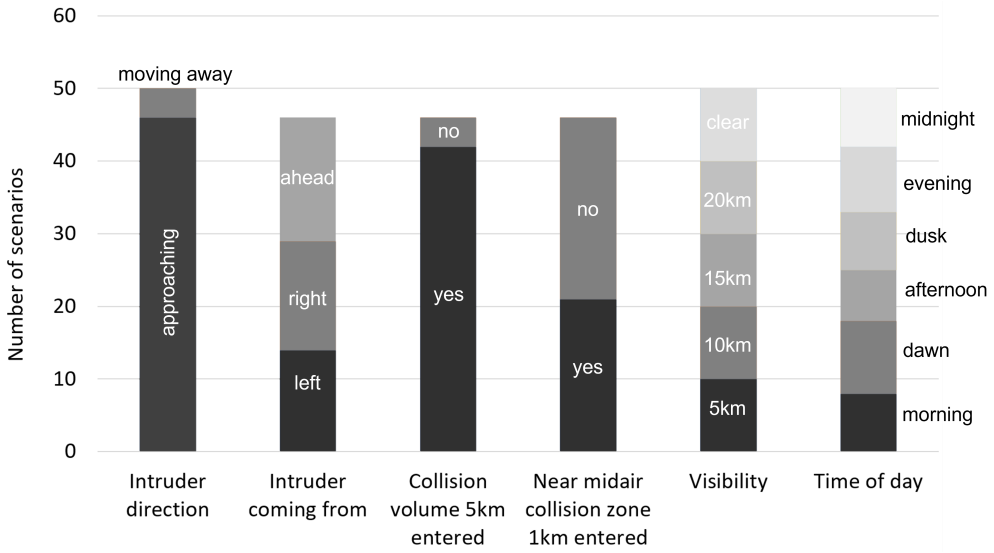

FIG 7. Data on 50 randomized collision avoidance scenarios generated by the framework for potential ODD coverage testing

6.3. Environmental Conditions

The operating parameters for the environmental conditions were limited to weather conditions like visibility, time of day, wind and rain, but for reasons of simplicity only the first two environmental conditions are demonstrated. The test cases were automatically recorded and stored with the corresponding ID together with the other logged data.
In Figure 8 visual examples of the environment out of the first ten test cases are depicted. On the upper side three pictures of daytime scenery's are depicted. On the lower side three night time scenarios are shown. From left to right the visibility in the scenarios decreases.
This visual data set is not highly relevant for demonstrating ODD coverage using an ADS-B sensor, however later in the project in which this paper was conducted a collision avoidance system based on image data might be implemented. In that case a wide ranging data set can help in the ODD coverage process.

7. CONCLUSION AND OUTLOOK

When it comes to AI engineering especially in safety critical domains such as aviation, developers encounter novel challenges in making sure the system operates as intended. According to EASA taking the ConOps and describing the operational scenarios provide engineers the basis to formulate an initial ODD which sets the boundary line of the operation. The paper aimed at creating a methodical approach for this process based on an exemplary use case of collision avoidance. Further, it was investigated how ODD could be either refined iteratively or tested for coverage of the operating parameters. Within the framework the initial ODD is exported to a python script-based scenario generator, that takes the ODD values for the creation of variable randomized scenarios. The python script generates scenario files for the open source simulator FlightGear, were the scenario execution takes places. Here a second python script comes into play and simulates the ODD monitoring part. The storing of the data also happens automated. Consequently, this framework systematically archives scenario information in both textual and image formats, which can be used for training and validating AI models to enhance scenario coverage in accordance with ODD descriptions. However, the evaluation is not fully automated yet, what will be crucial in the future if the number of operating parameters increase towards more realistic representations of the real world. Yet even with a limited number of operating parameters the general functionality of the framework could be demonstrated. Depending on the input for the initial ODD the framework provides data, that can be used for getting insights into the system's limitation and constraints especially on the edge cases. This work aims at describing an engineering approach that outlines the steps from ConOps towards ODD and testing ODD coverage. Considering the novelty of the ODD description concept for highly automated aircraft, it is necessary to assess the system capabilities including AI components to ensure the safe deployment of aircraft for a specific application. Future work aims to automate the framework entirely for scenario based-testing from ODD description. Additional efforts will be directed towards refining critical attributes and parameters that influences system functionality, thereby exploring critical scenarios. Moreover, this framework also explores critical and feasible parameters for AI-based maneuver prediction system, incorporating ADS-B sensor and camera sensor data into the scenario-based testing.

Contact address:

thomas.stefani@dlr.de

References

[1] Ramgopal Kashyap. Artificial intelligence systems in aviation. pages 1–26, 01 2019. ISBN: 9781522575894. DOI: 10.4018/978-1-5225-7588-7.ch001.

[2] Guillaume Soudain. Easa concept paper: First usabele guidance for level 1 & 2 machine learning applications: A deliverable of the easa ai roadmap, 2023. https://www.easa.europa.eu/en/downloads/137631/en.

[3] BSI Group. PAS 1883:2018 - Assuring the safety of automated vehicle trials and testing - Code of Practice. https://www.bsigroup.com/en-GB/automotive/publications/PAS-1883-2018/, 2018. [Online; accessed April 12, 2023].

[4] SAE. Taxonomy and definitions for terms related to driving automation systems for on-road motor vehicles,

FIG 8. Randomization of environmental conditions for collision avoidance scenario

2021. https://www.sae.org/standards/content/j3016_202104/.

[5] Ian Colwell. Runtime restriction of the operational design domain: A safety concept for automated vehicles. Master's thesis, University of Waterloo, 2018.

[6] Chen Sun. *Operational Design Domain Monitoring and Augmentation for Autonomous Driving*. PhD thesis, 2022. http://hdl.handle.net/10012/18964.

[7] Akshay Anilkumar Girija, Thomas Stefani, Ryan Mut, Thomas Krüger, and Umut Durak. Using operational design domain for safe ai in urban air mobility. In *Proc. ASAM International Conference 2022)*, Dresden, Germany, 2022.

[8] FlightGear developers & contributors. Flightgear. https://www.flightgear.org/.

[9] Hendrik Weber, Julian Bock, Jens Klimke, Christian Roesener, Johannes Hiller, Robert Krajewski, Adrian Zlocki, and Lutz Eckstein. A framework for definition of logical scenarios for safety assurance of automated driving. *Traffic injury prevention*, 20(sup1):565–570, 2019.

[10] Jasper Sprockhoff, Bojan Lukic, Vincent Janson, Alexander Ahlbrecht, Umut Durak, Siddhartha Gupta, and Thomas Krueger. Model-based systems engineering for ai-based systems. In *AIAA SCITECH 2023 Forum*, pages 1–4, 2023.

[11] Hrvoje Belani, Marin Vukovic, and Željka Car. Requirements engineering challenges in building ai-based complex systems. In *2019 IEEE 27th International Requirements Engineering Conference Workshops (REW)*, pages 252–255. IEEE, 2019.

[12] Ieee guide for information technology - system definition - concept of operations (conops) document. *IEEE Std 1362-1998*, pages 1–24, 1998. DOI: 10.1109/IEEESTD.1998.89424.

[13] Ahmed Irfan, Kyle D. Julian, Haoze Wu, Clark Barrett, Mykel J. Kochenderfer, Baoluo Meng, and James Lopez. Towards verification of neural networks for small unmanned aircraft collision avoidance. In *2020 AIAA/IEEE 39th Digital Avionics Systems Conference (DASC)*, pages 1–10, 2020. DOI: 10.1109/DASC50938.2020.9256616.

[14] Kyle D. Julian and Mykel J. Kochenderfer. Guaranteeing safety for neural network-based aircraft collision avoidance systems. In *2019 IEEE/AIAA 38th Digital Avionics Systems Conference (DASC)*. IEEE, sep 2019. DOI: 10.1109/dasc43569.2019.9081748.

[15] Nicola Kolb, Claudius Jordan, Florian Huber, and Alexander Pretschner. Automatic evaluation of automatically derived semantic scenario instance descriptions. In *2022 IEEE 25th International Conference on Intelligent Transportation Systems (ITSC)*. IEEE, 2022. DOI: 10.1109/itsc55140.2022.9922013.

[16] Tabea Schmidt and Alexander Pretschner. Stellauav: A tool for testing the safe behavior of uavs with scenario-based testing (tools and artifact track). In *2022 IEEE 33rd International Symposium on Software Reliability Engineering (ISSRE)*, pages 37–48, 2022. DOI: 10.1109/ISSRE55969.2022.00015.

[17] Andrea Piazzoni, Jim Cherian, Mohamed Azhar, Jing Yew Yap, James Lee Wei Shung, and Roshan Vijay. Vista: a framework for virtual scenario-based testing of autonomous vehicles. In *2021 IEEE International Conference on Artificial Intelligence Testing (AITest)*, pages 143–150, 2021. DOI: 10.1109/AITEST52744.2021.00035.

[18] Sajad Khatiri, Sebastiano Panichella, and Paolo Tonella. Simulation-based test case generation for unmanned aerial vehicles in the neighborhood of real flights. In *2023 IEEE Conference on Software Testing, Verification and Validation (ICST)*, pages 281–292, 2023. DOI: 10.1109/ICST57152.2023.00034.

[19] Demin Nalic, Tomislav Mihalj, Arno Eichberger, Maximilian Bäumler, and Matthias Lehmann. Scenario based testing of automated driving systems: A literature survey. *FISITA World Congress 2021 - Technical Programme*, 2021.

[20] Sven Hallerbach. *Simulation-based testing of cooperative and automated vehicles*. PhD thesis, 2015. https://oops.uni-oldenburg.de/id/eprint/4649.

[21] Patrick Weissensteiner, Georg Stettinger, Siddartha Khastgir, and Daniel Watzenig. Operational design domain-driven coverage for the safety argumentation of automated vehicles. *IEEE Access*, 11:12263–12284, 2023. DOI: 10.1109/ACCESS.2023.3242127.

[22] ASAM. ASAM OpenODD: Concept Paper, Version 1.0. https://www.asam.net/project-detail/asam-openodd-v100/, October 1 2021. [Online; accessed April 12, 2023].