Formally Bounding UAS Behavior to Concept of Operation with Operation-Specific Scenario Description Language

Christoph Torens* Umut Durak† Florian Nikodem‡ Sebastian Schirmer§
German Aerospace Center (DLR), Institute of Flight Systems, Braunschweig, Germany

Previous work introduced an approach for formally describing the concept of operations for unmanned aircraft. For this purpose, an existing language for simulation scenario description was adapted. In the context of the specific operation category, an upcoming European regulation for the operation of unmanned aircraft, the description and acceptance of the concept of operations plays a major role for flight approval on a per mission basis. This paper extends the previous approach further with combining the formalized description of the concept of operations with our existing approach for runtime monitoring. Monitoring the behavior at runtime can be used to enforce certain limits on the behavior. Therefore, the concept of operations is an ideal input for the monitoring approach. As a basis for the information relevant for the concept of operations the official annex to the guidelines document for the specific operation risk assessment is used, as well as an internal concept of operations document for a DLR research unmanned aircraft system.

I. Introduction

Recent regulatory efforts from EASA introduced the new concept of a specific category that allows a stepwise adaptation of certification requirements [1–3], Fig. 1. Based on a specific operation risk assessment, this category enables new aircraft system architectures and mission designs. The concept allows for operational restrictions, defined in the concept of operations, to maintain overall operation safety [4]. Therefore, the description of the concept of operations is one of the key elements for the approval of a Unmanned Aircraft System (UAS) operation.

There are many use cases for a formalized version of the Concept of Operation (ConOps) description, such as tool support for development of the ConOps document, simulation and validation [5]. One particular use case we will discuss in this paper is to use the formalized ConOps description, specifically the operational restrictions defined therein, as input for the automatic supervision of the operation.

The American Institute of Aeronautics and Astronautics Modeling and Simulation Technical Committee recently launched a working group towards the development of a standard scenario definition language for aviation. In previous work [5], this language specification was extended in order to model specific operation safety boundaries of a system. The goal of our previous research is to create a dialect of the simulation scenario definition language that can be used to describe safe scenarios for a particular system of interest. We further discussed the challenges and benefits of such a formalization of the concept of operations by adapting the simulation scenario definition language.

Traditionally, monitoring is used as a form of verification and to support debugging [6–8]. Limiting or bounding of behavior using runtime monitoring has been identified as a topic of interest [9, 10]. In particular, a standard has been developed in order to specify such aspects of runtime monitoring. This makes it possible to put guarantees on complex algorithms that would otherwise be hard to verify [11, 12].

The concept of a ConOps description is used in several use cases and domains. The following definitions can be found in literature:

A ConOps describes how the system will be operated during the life-cycle phases to meet stakeholder expectations. It describes the system characteristics from an operational perspective and helps facilitate an understanding of the system goals [13]

ConOps can be developed in many different ways, but usually share the same properties. In general, a ConOps will include a statement of the goals and objectives of the system; strategies, tactics, policies, and constraints affecting the

*Research Scientist, Unmanned Aircraft, AIAA Senior Member.
†Research Scientist, Flight Dynamics and Simulation, AIAA Senior Member.
‡Research Scientist, Safety Critical Systems and Systems Engineering.
§Research Scientist, Unmanned Aircraft, AIAA Member.

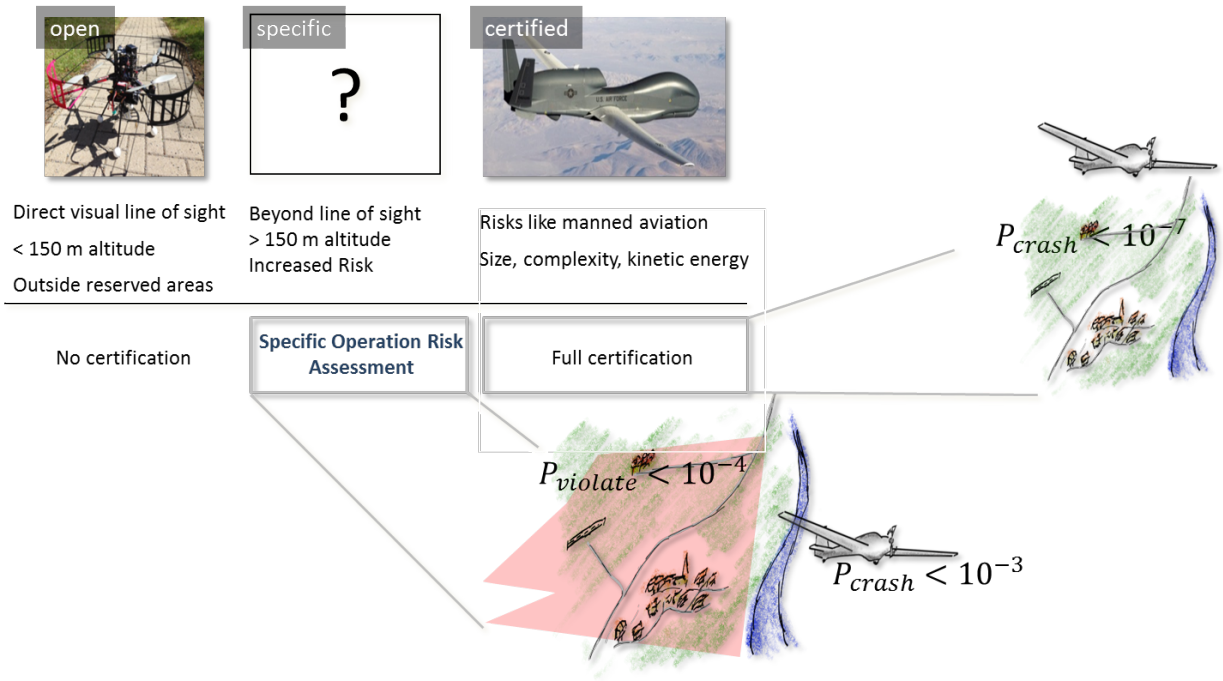


Fig. 1 Graphical explanation of the EASA categories for the operation of drones.

system; organizations, activities, and interactions among participants and operators; and operational processes for fielding the system [14]

The formalization of information to enable automated processing is a challenge for almost every IT project. Requirements engineering and modeling are topics constantly facing this challenge. The specific formalization of ConOps Document is shown in[14] to support a safety analysis. A ConOps document is formally defined, simply as a set of information:

$$\mathbb{C} = \{\bigcup_i I_i \mid (I_i \in \mathbb{I}) \wedge (I_i \cap I_j = \emptyset), \forall i \neq j; i, j \in \mathbb{N}\}$$

Each information element I is defined as a tuple (S,R,B,A) where:

- S is the source or subject of the information object I.
- R is the responsibility of the subject in control-theoretic terms.
- B is the type of behavior prescribed to the source.
- A is the context or set of assumptions, which provides the analysts' justification for assigning the first three elements to the quadruple I.

The elements of this tuple are further divided and characterized in more detail. However, in contrast to [14] we do not use the definition of a set and tuple to define ConOps and its information, but we define it as an instance of a formal language containing information as entities.

This remainder of this paper is organized as follows: after the motivation in this section and the related work on the topic, Section II describes the background and details of the information provided in a concept of operations document for the specific category. Section III goes over the details of the formalization approach using Operation-Specific Scenario Description Language (OSDL) that is based on System Entity Structure (SES) and XML Schema Definition (XSD). Section IV further details the approach of supervising the operation based on a formal technique of runtime monitoring. Section V describes how to combine these aforementioned approaches in order to achieve automated supervision of the ConOps restrictions and thusly, enabling safe operations. Finally, Section VI concludes this paper and gives an outlook on future work.

II. Overview of Concept of Operations

In the Specific Operations Risk Assessment (SORA) the ConOps is a crucial part of the assessment. The ConOps is the input to the SORA analysis and describes the intended operation, the UAS specifications and the operator's abilities to conduct such an operation. In the SORA process, both the operational description and the UAS specification is used to determine the operational ground and air risk. After the application of mitigations for both ground and air risk, a final Specific Assurance and Integrity Level (SAIL) is determined. The SAIL is linked to a set of operational safety objectives (OSO) that have to be fulfilled in order to receive an operation approval by the competent authorities. It is divided into six categories with increasing level of rigor, see Fig. 2 .

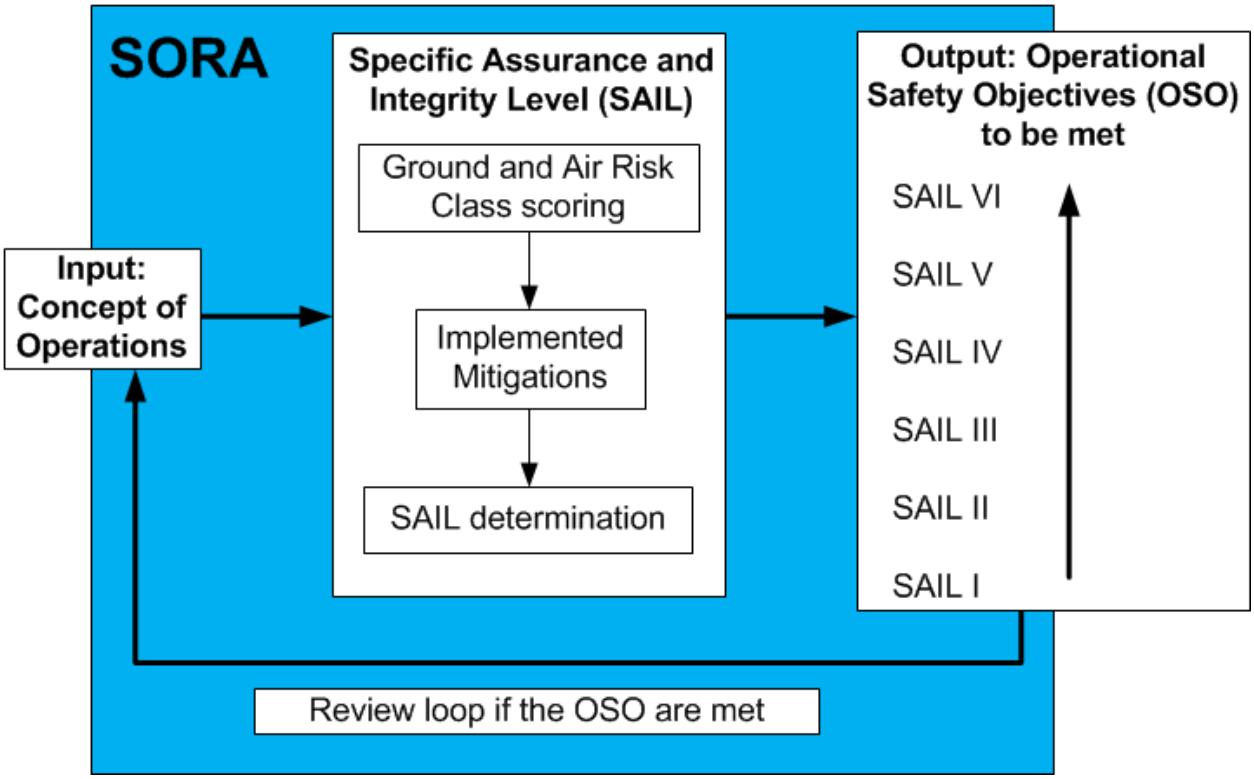


Fig. 2 Overview of the SORA process, described in [4].

In an iterative process, the mitigations, the OSO and evidence of their fulfillment must also be added to the ConOps. Interesting aspects in the scope of this paper are the implemented mitigations. These mitigations are further divided to specify those that describe risk reduction for people on ground and those for air traffic. The most interesting mitigation for the purpose of this paper is to assure containment for the area of operation and the operational airspace volume. This mitigation is meant to guarantee that the UAS remains in its approved operational volume given an additional risk buffer. As an example one can imagine an UAS operation that is conducted over a sparsely populated environment like a field or forest, but is also in the vicinity of a village or more populated environment. The containment mitigation ensures that the UAS is not able to leave its operational volume and reach the village. A technical solution that could help to achieve such containment could be a monitoring system that monitors the UAS status, position and heading. That monitoring system requires a kind of behavioral system model and some kind of knowledge of the route planning of the UAS. To simultaneously achieve the necessary information while remaining ConOps compliance, it seems mandatory to formalize the relevant parts of the ConOps. As stated above, the ConOps requires information on the UAS specifications, such as system architecture. It also has to contain operational information, for example the boundaries of the operational volume or even an explicit flight path, if necessary. In this paper we show a formalism approach of the aforementioned parts of the ConOps to assure containment and stay in compliance with the ConOps while conducting the operation.

Fig. 2 shows the importance of the ConOps in the SORA process. It is used to determine the SAIL and, therefore, the level of rigor of the threat barriers. Threat barriers represent requirements for the UAS and the operator to be met and are the output of the whole SORA process. After the SAIL is determined, the ConOps has to be reviewed by the

operation organization as well as the competent authority to ensure the threat barrier requirements are also met by the ConOps. If that is the case operation permission is given, otherwise the ConOps has to be reworked and the SORA process must be repeated again.

A. ConOps Information on Operational Restrictions

Relevant parameters in the ConOps document for monitoring are the UAS flight envelope, the environmental conditions and a map including operational volume boundaries and intended flight path. The UAS flight envelope resembles flight critical data from the legal specifications as well as important restrictions on environmental conditions as shown in Fig.3.

- Vehicle specifications
  - Maximum UAS velocity
  - Maximum UAS altitude
  - Maximum angle of attack
  - *Maximum attitude*
  - Maximum pushing angle
  - Maximum UAS mass
  - Center of gravity
- Environmental conditions
  - Weather conditions like rain, snow etc.
  - Operational temperatures
  - Wind conditions

Fig. 3 Relevant information and parameters from ConOps document.

The operational boundaries and flight path can be set as coordinates or something similar. It is important to keep in mind that an accurate interpretation of coordinate tables is hardly possible for humans. Therefore we recommend to use coordinate tables for formalization and to use additional pictures with operational volume boundaries and intended flight path for human interpretation. We highlighted the maximal attitude as an example that will be used throughout the paper.

B. ConOps Structure Analysis

As stated above, the ConOps should answer the questions that arise while perform the SORA process. The general structure of an appropriate ConOps document is not standardized. However, JARUS has published an annex to the SORA process. This annex serves as a guide in order to collect and present system and operational information for a UAS operation. In the guidance material the ConOps is structured as shown in Fig.4.

This is a rough example of the proposed general ConOps structure. The shown bullet points are further divided. Certainly when developing a ConOps for UAS use cases, information and important data on the same subject are spread out over the whole document. For example, the list above shows “Training of staff involved in operations” and “Training” in two separate locations. In order to support formalization and in addition create an easier to read document, we would recommend the concentration of all UAS relevant data in the “UAS Description” section and operational information such as flight path coordinates in the “Operations” section. Environmental conditions can either be meant as restriction of the UAS’ capabilities or as restrictions of the operation regardless of the UAS specifications. The operator or the competent authority might regard an operation conducted at certain weather conditions as too dangerous, even if the UAS is capable of these due to specification. Such information should be placed in the “Operations” section. The environmental conditions that focus on the UAS, e.g. UAS operation in heavy rain conditions is impossible, should be placed in the “UAS Description” section. In the following we show an approach how to formalize the important sections of the ConOps for monitoring and how these sections can be written to support the formalization.

III. Formalizing Concept of Operation

As discussed in previous section, ConOps provides the informal specification of the systems and its operation. It not only includes normal, but also introduces the abnormal. Thereby, it implicity presents the constraints that ensure safe operation. We proposed previously using ConOps to design a scenario description language that will ensure specification of simulation scenarios that conform to the constraints of ConOps [5].

A simulation scenario is defined as the specification of initial and terminal conditions, significant events and the environment as well as the major entities, their capabilities, behavior and interactions over time [15]. Despite the fact that the importance of simulation scenarios has long been well-known, there still exists a lack of common understanding

- Operation relevant information
  - Organization overview
    - Safety
    - Design and Production
    - Training of staff involved in operations
    - Maintenance
    - Crew
    - UAS Configuration Management
    - Other positions and other information
  - Operations
    - Type of operations
    - Standard Operating Procedures
    - Normal Operation Strategy
    - Abnormal operation and emergency operation
    - Accidents, incidents and mishaps
  - Training
    - General information
    - Initial training and qualification
    - Procedures for maintenance of currency
    - Flight Simulation Training Devices
    - Training program
- Technical relevant information
  - UAS Description
  - UAS Control segment
    - General
    - Navigation
    - Autopilot
    - Flight Control System
    - Control Station
    - Detect And Avoid System
  - Geo-fencing
  - Ground Support Equipment segment
  - Command and Control (C2) Link segment
  - C2 Link degradation
  - C2 Link Lost
  - Safety features

Fig. 4 Proposed ConOps structure.

and standardized practices in the aviation domain. This eventually leads to degraded interoperability of simulators and shareability of simulation scenarios. To address these issues, the American Institute of Aeronautics and Astronautics (AIAA) Modeling and Simulation Technical Committee (MSTC) launched a working group towards the development of a standard simulation scenario definition language for the aviation domain [16]. Within the scope of this effort, Durak et al. proposed using System Entity Structure for specifiying a domain specific languge for simulation scenario definition [17].

SES is a high-level ontology which was introduced for knowledge representation of decomposition, taxonomy and coupling of systems [18]. This formal ontology framework is used to define data engineering ontologies [19]. It is axiomatically defined to represent the elements of a system (or physical environment) and their relationships in a hierarchical manner. SES is a declarative knowledge representation scheme that characterizes the structure of a family of models in terms of decompositions, component taxonomies, and coupling specifications and constraints.

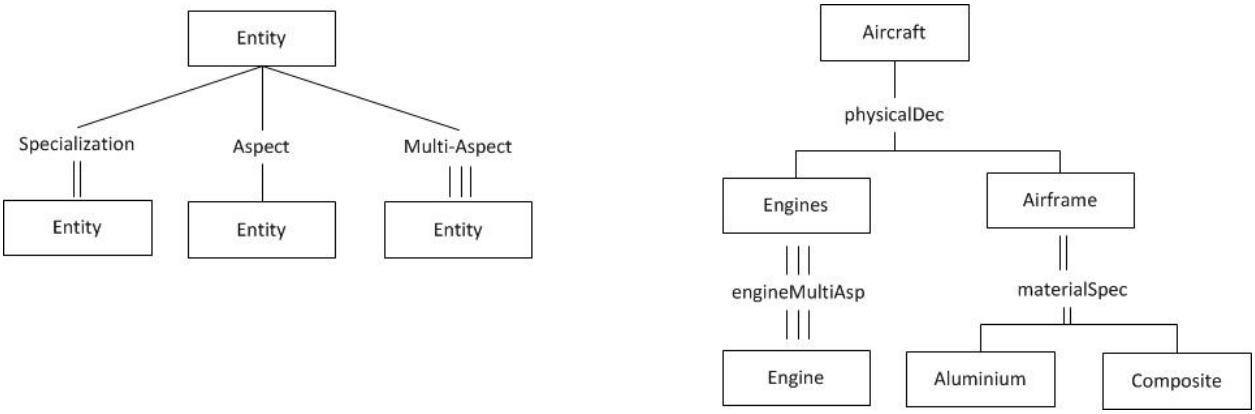


Fig. 5 SES Elements (left) and An Example (right).

Fig.5 provides a quick overview of the nodes and relationship involved in a SES. Entities represent things that have existence in a certain domain. They can have variables which can be assigned a value within given range. An Aspect

expresses a way of decomposing an object into more detailed parts and is a labeled decomposition relation between the parent and the children objects. Multi-Aspects are aspects for which the components are all of the same kind. A Specialization represents a category or family of specific forms that a thing can assume. It is a labeled relation that expresses alternative choices that a system entity can take on.

Pruning is defined as assigning the values to the variables and resolving the choices in Aspect, Multi-Aspect and Specialization relations. The result is Pruned Entity Structure (PES), which is a selection-free tree.

Durak et al. [20] propose an SES-based metamodeling approach for simulation scenario definition language. The SES metamodel captures all SES constructs and their relationships. Conforming to the SES metamodel, the scenario metamodel is defined to capture all possible elements of scenarios. Finally, scenario modeling is a pruning operation which prunes the scenario metamodel that is formalized as an SES. The computational representation of scenario metamodel is using XSD. And a particular scenario is then an XML document that conforms to the XML Schema of the simulation scenario definition language.

In our previous study [5], we prosed to extend the simulation scenario definion language. This is done through a formal specification of the ConOps constraints. An emphasis is placed on the scenario elements, specifically by constraining the ranges of scenario element attribute types. The computational representation of the ranges were implemented using the XML Schema restriction element. We called the metamodel specification for a scenario definition language that is constraint by a specific ConOps as Operation-Specific Scenario Description Language. An OSDL for a particular UAS is specified based on its ConOps using SES. Its computational representation is an XSD file. The scenario for an individual mission of that UAS is then specified using prunning. The scenario for that mission is an XML document that conforms to it schema definition, or in other words, its OSDL. Thereby, we guarantee that the scenario conforms to the constraints that we captured in OSDL.

This provided us with a means of defining a scenario formally and check if the UAS in its initial conditions and at any operation conform to ConOps constraints, but it was not possible to check and supervise the constraints at runtime. This paper investigates the utilization of a runtime monitoring together with a scenario definition language. Thereby, it is possible to ensure the conformance of the UAS to the constraints of the ConOps during its operation.

IV. Runtime Monitoring

Runtime monitoring is a lightweight formal method. Similar to exhaustive formal methods, such as model checking or automated theorem proving, a lightweight formal method has a formal basis but is limited in its scope, e.g. partiality in language or partiality in analysis [21]. Specifically, runtime monitoring does not evaluate all possible system executions but instead it evaluates the current execution. The approach is depicted in Figure 6.

Given a formal specification of the desired system behavior, a monitor is automatically generated which evaluates the sequence of given system events. The verdict returned by the monitor depicts either a violation of the specification or the adherence of the system to the specification. Also, the verdict of the monitor can be fed back to the system such that the system can react upon it. In order to receive events, an instrumentation of the system is often required. In [6–8], three essential aspects for a monitor integration into a system are listed, namely: realizability, responsiveness, and unobstrusiveness. The approach scales to more complex practical systems for which exhaustive formal methods often turn out to be infeasible. Further, since we evaluate the specification along the system execution, more expressive properties can be specified, e.g. statistical average. However, the evaluation and generation of verdicts are done at runtime and not at design-time. The main concerns of runtime monitoring are efficient and correct generated monitors, and the expressiveness of the specification language. In general, runtime monitoring

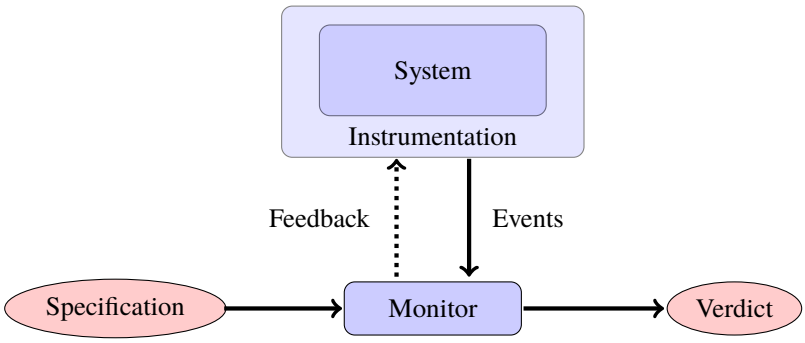


Fig. 6 Runtime Monitoring: Given a formal specification of the desired system behavior, a monitor is generated which evaluates the sequence of system events, i.e. the execution trace of the system.

complements exhaustive formal methods. During development, runtime monitoring can supplement conventional testing or debugging since it is significantly more powerful and versatile. After deployment, runtime monitoring can be used to ensure reliability, safety, and security of the system. Due to their formal basis, guarantees on the monitors can be given, e.g. memory bounds, which are desirable for safety-critical systems.

One specification language for runtime monitoring is called Lola [22]. Lola describes an equation system and is evaluated in a synchronous manner. To illustrate its usage, consider the following classic Lola specification*:

input int *velocity*
input bool *enabled*
output double *avgWindow* := if *enabled* { (*velocity*[-2,0] + *velocity*[-1,0] + *velocity*) / 3.0 } else { 0.0 }
trigger *avgWindow* > 10.0 with "VIOLATION: *avgWindow* exceeds the threshold!"

The given Lola specification states that two input streams are expected: the integer stream *velocity* and the boolean stream *enabled*. Given these input stream, the generated Lola monitor evaluates the output stream *avgWindow*. Specifically, in each execution step the monitor checks whether its input *enabled* is $true$ at the current trace position. If so, the output *avgWindow* represents the average over the last three received *velocity* values, otherwise *avgWindow* evaluates to 0.0. The expression $s[x, y]$ allows to access previous and future values of stream s. The variable x represents the position offset where negative values indicate past accesses. Further, variable y states the out-of-bounds value which is required when accessing a trace position which does not exist, e.g. being at the first trace position and accessing a previous one. A $trigger$ represents a notification to the user that the condition, here $avgWindow > 10$, evaluated to $true$.

V. Enforcing ConOps Restrictions with Runtime Monitoring

Given the Lola specification language and its capabilities, we are able to take the formal definition of the CONOPS document and provide a means to formally verify that the operational restrictions hold. The approach is depicted in Fig.7. In the following, we will give a very simple example transformation.

As detailed in section III, OSDL is used to describe the operational restrictions. OSDL itself is described by an XML Schema definition. The elements of the scenarios that are captured include environment, entities such as the aircraft with its systems, airport, and events. Attributes are then used to define the scenario parameters. The details of the language are presented in recent papers[24, 25] of Durak and colleagues. To improve the understanding of a formalized ConOps, a representative excerpt of an OSDL is given in Fig.8.

The given excerpt encompasses the description of the used aircraft, e.g. the flight descripton with its maximal values for `Position`, `Attitude`, `AngularVelocity`, and `TranslationalVelocity`. Figure 8 zooms into the maxmial `Attitude` and `AngularVelocity`. Further, on the right, the corresponding XML schema is depicted.

Given that the CONOPS document has this formal definition, it is possible to automate this process and provide a means to formally verify the information requirements.

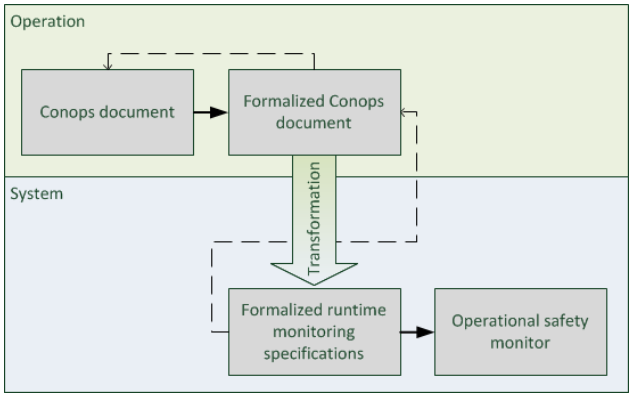


Fig. 7 Transformation from formalized ConOps description (Operation layer, green) to formal monitoring specifications (System layer, blue).

The information available in the XML schema are used to infer the corresponding formal specification to automatically generate monitors guaranteeing the adherence of the operation to the ConOps. A Lola specification corresponding to

*There are several extensions: Lola2.0 [23] extends Lola by parametrization and dynamic stream creation; RTLola offers real-time offsets and is capable to handle asynchronous behavior (to be published).

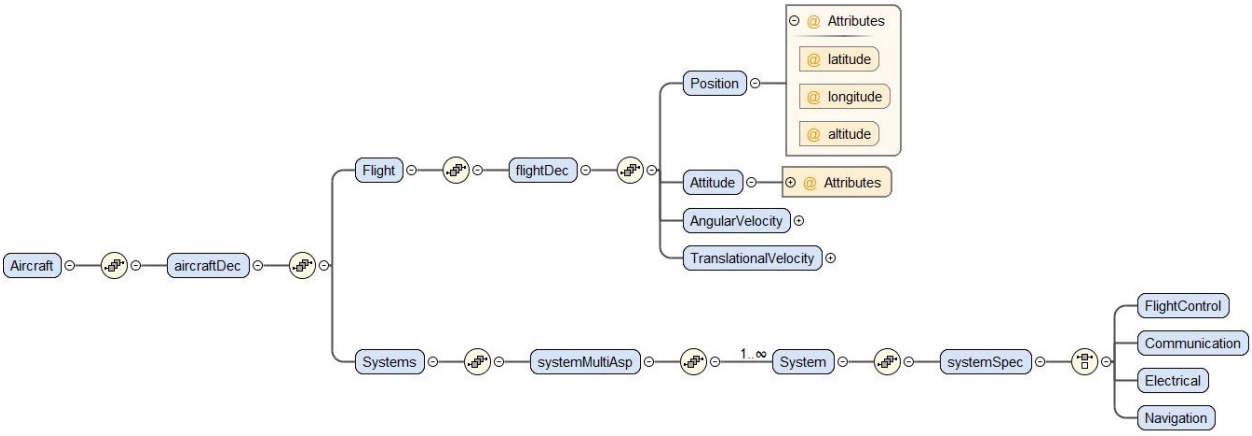
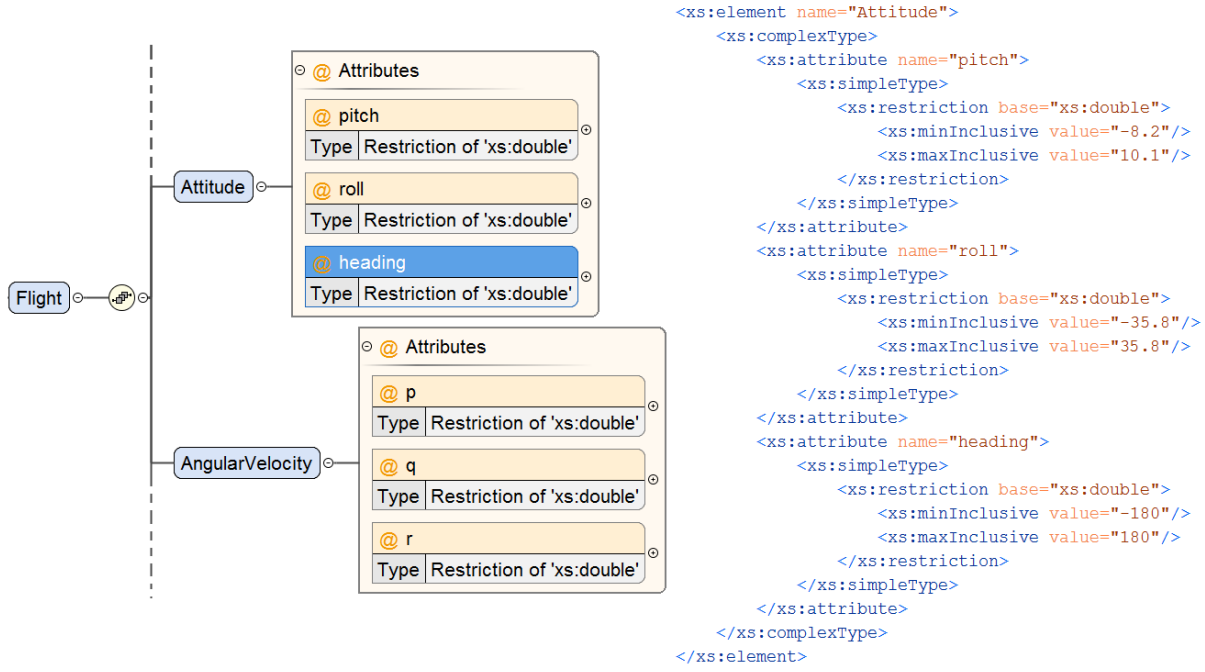


Fig. 8 An Excerpt from the Simulation Scenario Definition XML Schema.




```
<xs:element name="Attitude">
    <xs:complexType>
        <xs:attribute name="pitch">
            <xs:simpleType>
                <xs:restriction base="xs:double">
                    <xs:minInclusive value="-8.2"/>
                    <xs:maxInclusive value="10.1"/>
                </xs:restriction>
            </xs:simpleType>
        </xs:attribute>
        <xs:attribute name="roll">
            <xs:simpleType>
                <xs:restriction base="xs:double">
                    <xs:minInclusive value="-35.8"/>
                    <xs:maxInclusive value="35.8"/>
                </xs:restriction>
            </xs:simpleType>
        </xs:attribute>
        <xs:attribute name="heading">
            <xs:simpleType>
                <xs:restriction base="xs:double">
                    <xs:minInclusive value="-180"/>
                    <xs:maxInclusive value="180"/>
                </xs:restriction>
            </xs:simpleType>
        </xs:attribute>
    </xs:complexType>
</xs:element>
```

Fig. 9 OSDL Approach

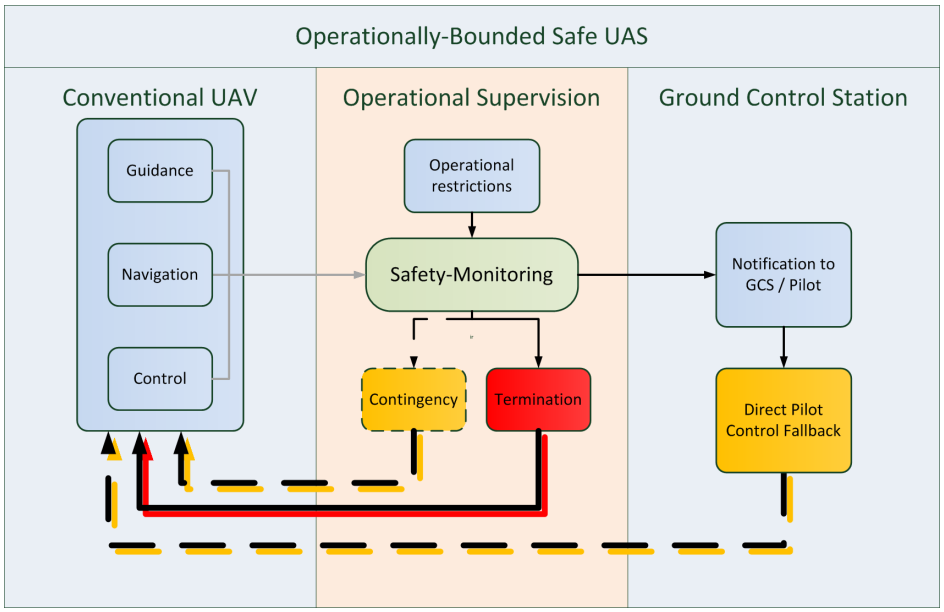


Fig. 10 Overview of monitoring architecture for the operationally-bounded safe UAS.

the except given in Fig. 9 is:

input double *pitch*, *roll*, *heading*
output double *checkPitch* := -8.2 <= *pitch* <= 10.1
output double *checkRoll* := -35.8 <= *roll* <= 35.8
output double *checkHeading* := -180 <= *heading* <= 180
trigger *!* *(checkPitch & checkRoll & checkHeading)* with "VIOLATION: Limits are exceeded!"

The monitors can be used to have an operationally-bounded safe UAS depicted in Fig.10. The figure shows an UAS consisting of an Unmanned Aerial Vehicle (UAV) and a ground control station (GCS). However, as an additional layer that provides the operational supervision to enable operational safety. This layer performs the runtime monitoring approach described in Section IV. If a violation of any operational restrictions is detected, the safety monitor triggers one of three possible actions.

- Informing GCS and a safety pilot about the current system health, e.g. degradation or limit violations. Due to the physical distance to the UAS, these information help the GCS or the safety pilot to improve their situational awareness and, therefore, helps to enforce ConOps boundaries. Degradation reports or in general warning prior to the violation are of course preferable since a limit violation can be prevented.
- Activating a contingency procedure, possibly preventing a violation of the operational restrictions. For operations beyond visual line of sight of a safety pilot, a possible use-case is geo-fencing. Geo-fencing represent virtual fences bounding the operation area. Having a contingency procedure enforcing the geo-fencing is a central aspect for safe flights beyond visual line of sights. A monitor can be used as a sophisticated option activating a contingency to prevent the UAV to break out of the geo-fence.
- Termination of the aircraft, if all of the above options are not applicable or did not result in a beneficial outcome. This option can be used as a last resort to ensure operational safety and prevent the UAV from violating operational restrictions.

VI. Conclusion and Future Work

In this paper we showed how a formalized description of the concept of operations for UAS can be combined with an approach for the automated supervision of the UAS operation. The restriction of the operation is a key concept of the

specific category. The concept of operations requires detailed information regarding operation, vehicle parameters, and operational restrictions. This combination of automated supervision of the operational restrictions is a perfect use case for the formalized description of the ConOps document. In fact, according to the ConOps document, the strict application of this approach can automatically ensure the safety of the operation. This use case is arguably enough motivation to utilize and enable a formal description of the ConOps document, but there are other use cases that have a similar impact on the specific category. These are mentioned in this paper and previous work. This paper demonstrates the use case of combining the formal ConOps description with our approach on runtime monitoring. However, for a real-world application there are still necessary developments that need to be made in order to enable such an use case. In particular:

- The improvement of tool support for the formalization and development of the ConOps description.
- A further synchronization between operational restrictions and supervision capabilities.
- The development of automated transformations from ConOps description to formal monitoring specifications, incorporating calculations for complex consistency checks, such as geofence safety buffers.

Future work will concentrate on the improvement of these aspects and more.

References

[1] European Aviation Safety Agency, "Concept of Operations for Drones, A risk based approach to regulation of unmanned aircraft," *EASA*, 2015.

[2] European Aviation Safety Agency, "Advance Notice of Proposed Amendment 2015-10, Introduction of a regulatory framework for the operation of drones," *EASA*, 2015.

[3] European Aviation Safety Agency, "Introduction of a regulatory framework for the operation of drones, Advance Notice of Proposed Amendment 2017-05," *EASA*, 2017.

[4] Joint Authorities for Rulemaking of Unmanned Systems, "JARUS Guidelines on Specific Operations Risk Assessment (SORA), Draft for public consultation, V0.2 Joint Authorities for Rulemaking of Unmanned Systems," *JARUS*, 2016.

[5] Torens, C., Durak, U., Nikodem, F., Dauer, J. C., Adolf, F.-M., and Dittrich, J. S., "Adapting Scenario Definition Language for Formalizing UAS Concept of Operations," *AIAA Modeling and Simulation Technologies (MST) Conference. AIAA Modeling and Simulation Technologies Conference, 8.-12. Jan. 2018, Kissimmee, FL, USA. DOI: 10.2514/6.2018-0127*, 2018, pp. 1–8.

[6] Geist, J., Rozier, K. Y., and Schumann, J., "Runtime Observer Pairs and Bayesian Network Reasoners On-board FPGAs: Flight-Certifiable System Health Management for Embedded Systems," *Runtime Verification - 5th International Conference, RV 2014, Toronto, ON, Canada, September 22-25, 2014. Proceedings*, Lecture Notes in Computer Science, Vol. 8734, edited by B. Bonakdarpour and S. A. Smolka, Springer, 2014, pp. 215–230. URL `http://dx.doi.org/10.1007/978-3-319-11164-3_18`.

[7] Reinbacher, T., Rozier, K. Y., and Schumann, J., "Temporal-Logic Based Runtime Observer Pairs for System Health Management of Real-Time Systems," *Tools and Algorithms for the Construction and Analysis of Systems - 20th International Conference, TACAS 2014, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2014, Grenoble, France, April 5-13, 2014. Proceedings*, Lecture Notes in Computer Science, Vol. 8413, edited by E. Ábrahám and K. Havelund, Springer, 2014, pp. 357–372. URL `http://dx.doi.org/10.1007/978-3-642-54862-8_24`.

[8] Schumann, J., Moosbrugger, P., and Rozier, K. Y., "R2U2: Monitoring and Diagnosis of Security Threats for Unmanned Aerial Systems," *Runtime Verification - 6th International Conference, RV 2015 Vienna, Austria, September 22-25, 2015. Proceedings*, Lecture Notes in Computer Science, Vol. 9333, edited by E. Bartocci and R. Majumdar, Springer, 2015, pp. 233–249. URL `http://dx.doi.org/10.1007/978-3-319-23820-3_15`.

[9] Hook, L. R., Clark, M., Sizoo, D., Skoog, M. A., and Brady, J., "Certification strategies using run-time safety assurance for part 23 autopilot systems," *Aerospace Conference, 2016 IEEE*, IEEE, 2016, pp. 1–10.

[10] Skoog, M. A., Prosser, K., and Hook, L., "Ground collision avoidance system (iGCAS)," , Apr. 25 2017. US Patent 9,633,567.

[11] ASTM F38.01 UAS Airworthiness, "F3269-17 Standard Practice for Methods to Safely Bound Flight Behavior of Unmanned Aircraft Systems Containing Complex Functions," 2017.

[12] Cook, S. P., "An ASTM Standard for Bounding Behavior of Adaptive Algorithms for Unmanned Aircraft Operations," *AIAA Information Systems-AIAA Infotech@ Aerospace*, 2017, p. 0881.

[13] Kapurch, S. J., *NASA Systems Engineering Handbook*, Diane Publishing, 2010.

[14] Fleming, C. H., "Safety-driven early concept analysis and development," Ph.D. thesis, Massachusetts Institute of Technology, 2015.

[15] Topçu, O., Durak, U., Oğuztüzün, H., and Yilmaz, L., *Distributed simulation: A model driven engineering approach*, Springer, 2016.

[16] Durak, U., Jafer, S., Beard, S. D., Reardon, S., Murphy, J. R., Crider, D. A., Gerretsen, A., Lenz, H., Macchiarella, N. D., Rigby, K. T., et al., "Towards a Standardization for Simulation Scenario Development in Aviation-Panel Discussion," *2018 AIAA Modeling and Simulation Technologies Conference*, 2018, p. 1395.

[17] Durak, U., Pruter, I., Gerlach, T., Jafer, S., Pawletta, T., and Hartmann, S., "Using System Entity Structures to model the elements of a scenario in a research flight simulator," *AIAA Modeling and Simulation Technologies Conference*, 2017, p. 1076.

[18] Kim, T.-G., Lee, C., Christensen, E. R., and Zeigler, B. P., "System entity structuring and model base management," *IEEE Transactions on Systems Man and Cybernetics*, Vol. 20, No. 5, 1990, pp. 1013–1024.

[19] Zeigler, B. P., and Hammonds, P. E., *Modeling and simulation-based data engineering: introducing pragmatics into ontologies for net-centric information exchange*, Academic Press, 2007.

[20] Durak, U., Jafer, S., Wittman, R., Mittal, S., Hartmann, S., and Zeigler, B. P., "Computational Representation for a Simulation Scenario Definition Language," *AIAA Modeling and Simulation Technologies Conference*, 2018. doi:10.2514/6.2018-1398.

[21] Jackson, D., and Wing, J., "Lightweight Formal Methods," , Apr. 1996.

[22] D'Angelo, B., Sankaranarayanan, S., Sánchez, C., Robinson, W., Finkbeiner, B., Sipma, H. B., Mehrotra, S., and Manna, Z., "Lola: Runtime Monitoring of Synchronous Systems," *12th International Symposium on Temporal Representation and Reasoning (TIME'05)*, IEEE Computer Society Press, 2005, pp. 166–174.

[23] Faymonville, P., Finkbeiner, B., Schirmer, S., and Torfah, H., "A Stream-Based Specification Language for Network Monitoring," *Runtime Verification - 16th International Conference, RV 2016, Madrid, Spain, September 23-30, 2016, Proceedings*, Lecture Notes in Computer Science, Vol. 10012, edited by Y. Falcone and C. Sánchez, Springer, 2016, pp. 152–168. doi:10.1007/978-3-319-46982-9_10, URL http://dx.doi.org/10.1007/978-3-319-46982-9_10.

[24] Durak, U., Pruter, I., Gerlach, T., Jafer, S., Pawletta, T., and Hartmann, S., "Using System Entity Structures to Model the Elements of a Scenario in a Research Flight Simulator," *AIAA Modeling and Simulation Technologies Conference*, Grapevine, TX, 2017.

[25] Durak, U., Jafer, S., Wittman, R., Mittal, S., Hartmann, S., and Zeigler, B. P., "Computational Representation for a Simulation Scenario Definition Language," *AIAA Modeling and Simulation Technologies Conference*, Kissimmee, FL, 2018.