drones

MDPI

Article

Rule-Based Verification of Autonomous Unmanned Aerial Vehicles

Christoph Sieber *, Luis Miguel Vieira da Silva, Kilian Grünhagen and Alexander Fay

Institute of Automation Technology, Helmut Schmidt University/University of the Federal Armed Forces Hamburg, 22043 Hamburg, Germany; miguel.vieira@hsu-hh.de (L.M.V.d.S.); kilian.gruenhagen@hsu-hh.de (K.G.); alexander.fay@hsu-hh.de (A.F.)

* Correspondence: christoph.sieber@hsu-hh.de

Abstract: Automation enhances the capabilities of unmanned aerial vehicles (UAVs) by enabling self-determined behavior, while reducing the need for extensive human involvement. Future concepts envision a single human operator commanding multiple autonomous UAVs with minimal supervision. Despite advances in automation, there remains a demand for a "human in command" to assume overall responsibility, driven by concerns about UAV safety and regulatory compliance. In response to these challenges, a method for runtime verification of UAVs using a knowledge-based system is introduced. This method empowers human operators to identify unsafe behavior without assuming full control of the UAV. Aspects of automated formalization, updating and processing of knowledge elements at runtime, coupled with an automatic reasoning process, are considered. The result is an ontology-based approach for runtime verification, addressing the growing complexity of UAVs and the need to ensure safety in the context of evolving aviation regulations.

Keywords: automation; autonomy; drones; knowledge-based systems; monitoring; rules; runtime verification; safety; unmanned aerial vehicles

check for updates

Citation: Sieber, C.; Vieira da Silva, L.M.; Grünhagen, K.; Fay, A. Rule-Based Verification of Autonomous Unmanned Aerial Vehicles. *Drones* **2024**, *8*, 26. https://doi.org/10.3390/drones8010026

Academic Editor: Diego González-Aguilera

Received: 30 November 2023
Revised: 2 January 2024
Accepted: 15 January 2024
Published: 20 January 2024

1. Introduction

The use of unmanned aerial vehicles (UAVs) offers potential for a wide range of scenarios and applications. The exploitation of this potential is being driven forward by continuous innovation and has gained considerable momentum in recent years. One indicator of this development is the recent exponential growth in the published literature on UAVs [1]. In addition, forecasts for the market growth of UAVs, for example, in the European Union, indicate a significant increase, which underlines the growing importance of this technology in various sectors [2]. In particular, the use of automation technologies has a significant impact on the way UAVs are operated. On the one hand, this leads to a reduction in the amount of human workforce required ([3]); on the other hand, automation increases the capabilities of the UAVs themselves by enabling them to perform self-determined tasks ([4]). These developments peak in the vision of a single operator being able to command and control teams of multiple autonomous UAVs with minimal human supervision [5].

Despite these advances, however, there is still a demand for a "human in command" who assumes overall responsibility for the UAV [6]. However, the increasing complexity of UAVs and the associated difficulty in predicting their behavior raises concerns about whether humans can adequately take on this responsibility at all. Concerns about the safety of UAVs are a decisive influencing factor. This aspect becomes particularly important in the context of legal considerations, as the main objective of aviation regulations is to ensure an adequate level of safety [2]. There are already multiple approaches that can guarantee aspects of safety in different ways. For example, the approaches of [7,8] can monitor and control the altitude of a specific UAV even under adverse conditions. The commercially available FlySafe system ([9]), on the other hand, can effectively prevent access to geographical zones. However, these approaches cannot determine whether

the specified altitude value is appropriate or whether entering a specific zone is indeed prohibited in a particular situation. In addition, these approaches are only transferable to different types of UAVs to a limited extent.

Considering this background, this paper presents a method for rule-based runtime verification of UAVs using a knowledge-based system. This method enables a human operator to specifically formulate and detect unsafe behavior, taking into account the particular situation, without taking full control of the respective UAV.

The major contribution of this paper is the automated formalization, updating and processing of different elements of knowledge at runtime, as well as an automatic reasoning process based on this. This is used to create and exemplify an ontology-based approach for rule-based runtime verification.

This paper is organized as follows: Section 2 initially provides necessary background information on levels of automation and autonomy, followed by safety constraints, verification techniques and knowledge representation. Section 3 then introduces a method to create and process verification rules. In Section 4, an implementation of this method is demonstrated by means of a simple use case for traffic monitoring. Section 5 discusses the main results presented in this paper, highlighting predictability. Finally, a short conclusion is given in Section 6, as well as an outlook on further research.

2. Background Information

This section provides necessary background information on automated and autonomous UAVs and the demand for the safe behavior of such UAVs. Additionally, verification techniques are discussed with a special focus on rule-based verification. Finally, rule knowledge and the representation and processing of knowledge in knowledge-based systems are explained.

2.1. Automated and Autonomous Unmanned Aerial Vehicles

Automation and autonomy, though sometimes used synonymously, have different meanings, and thus an autonomous operation of a UAV may have differing requirements regarding certification, insurance coverage or pilot qualifications [6]. Autonomy is considered a key factor enabling UAVs to perform highly complex missions. It includes self-determined planning of actions as well as self-organized communication and coordination with other UAVs [4]. Nonetheless, less complex missions may also be usefully supported by automation techniques. The benefits of automation include the takeover of specific actions such as stabilization, the ease of planning, executing and repeating missions and the ability of a single human to command and control multiple UAVs in the role of a supervisor without needing the skills of a trained pilot [3].

There are various approaches to delineating the meaning of autonomy and automation in the field of robotics and UAVs. Established scientific approaches include the ALFUS framework ([10]), considering mission complexity, environmental difficulty and human independence, and the 10-point scale by Sheridan (adapted in [11]), considering the dependence of human decisions, leveling from total manual control to the complete ignorance of human decisions. A classification of automation levels for UAVs with a normative status was released by the European Cockpit Association (ECA). It is based on the established classification scheme for driving automation, defined in SAE J3016. In six so-called automation levels, ranging from “No Automation” to “Full Automation”, the ECA defines responsibilities for the management of flight [6]. A detailed legal classification scheme is lacking. In Europe, regulation (EU) no. 2018/1139 merely states within its definition of a remote pilot that during the automated flight of a UAV, the remote pilot remains able to intervene [12]. Regulation (EU) no. 2019/947 amends that during an autonomous operation of a UAV, the remote pilot is not able to intervene [13].

Regardless of the classification schemes chosen, it can be stated that autonomy is similar to higher levels of automation. With each additional level, the ability of the human operator to control the UAV decreases, while the performance of the automation increases.

The highest levels of automation or autonomy without the possibility of human interaction are also described in some schemes, but it remains uncertain whether these levels should really be reached, such as, for example, a level where the automated system ignores the human ([11], Level 10). The ECA demands that a human must always be in command [6], while [14] concludes that the legislator within [12] only states that a course change by the remote pilot can no longer take place. However, an interaction, e.g., in the form of an emergency stop, remains possible. Thus, the human may be restricted in their interaction possibilities with higher automation but always retains possibilities of command and responsibility for the UAV. This indispensable responsibility implies that the human must always be able to ensure that the UAV does not violate any constraints, especially in relation of safety—even with limited means of command. An enablement to determine any constraint violation is therefore essential.

Considering this ambivalence of safety and autonomy, the authors endorse the following definition of autonomy: "Autonomy is a state in which a robot system, once activated, is capable of autonomously performing some or all of the mission tasks, for a specified period of time or continuously, in specified areas or everywhere, while it must remain possible at all times for a technical supervisor to shut down the system to a low-risk, operable state in situations that cannot be foreseen by human judgement" (translated from [14]). It ensures human governance by setting up goals and an operational framework, while it gives the UAV the maximum possible self-determination to operate.

2.2. Safety Constraints in the Context of Unmanned Aerial Vehicles

Constraints are spaces or periods of time where the performance of one or more activities is restricted or prohibited. Together with goals and mission execution, they form the essence of a mission [15]. Using automated UAVs, it is possible to plan and execute missions for various scenarios and applications. Established software for so-called autopilots and control stations (e.g., px4 [16] in combination with QGroundControl [17]) allow the manual planning of waypoints and the automated, optionally repeatable, execution of these missions [18]. In such lower levels of automation, it may not yet be necessary to explicitly formulate constraints due to very good predictability. For instance, waypoints may simply be plotted around a restricted zone to reach a waypoint. Violation of the constraint is thus effectively prevented without the UAV having to be aware of the constraint "restricted zone" or even to understand it. Yet, this represents an ideal state, as missions planned by humans may contain errors.

However, this linear approach is not suitable for complex missions. Within higher levels of automation or autonomy, the UAV should be enabled to perform actions based on the current situation and resolve emerging conflicts on its own in order to achieve its goals [4]. For this, an understanding of these goals, as well as given constraints, is necessary in order to provide the UAV with a maximum spectrum of possible actions. A granular specification of how goals are to be achieved is no longer necessary.

Safety is a serious motivation for formulating constraints. It is defined as the freedom from risk, which is not tolerable, but is not to be mistaken as a guarantee of total freedom from any risk [19]. Thus, the absence of safety may result in hazard, danger or even harm. Since a key ability of UAVs is locomotion, traffic safety, a subarea of safety, is especially considered as significant. It refers to freedom from intolerable risks caused by motion [20]. Relevant safety constraints may, for example, be generated by conducting a safety assessment procedure or additionally be taken from normative or legally binding sources. A substantial number of such constraints for the traffic safety of UAVs is provided by the legislator in the form of legal regulations. These are either explicitly formulated for UAVs or implicitly formulated for aircraft in general [21]. A violation of such a safety constraint thus constitutes unsafe behavior. Conversely, this means that if absolutely no safety constraints are violated, the UAV behaves safely. However, the latter statement is only true if all safety constraints are fully known. This is a case that is difficult to prove,

especially when considering highly automated UAVs in open, uncontrolled environments with self-deterministic behavior and the ability to adapt to changing situations.

2.3. Verification of Self-Determined Behavior

Since autonomous UAVs may be used in various environments performing different scenarios while always exploiting a maximum range of actions as part of their mission execution, commonly used techniques, such as testing and simulation, are not sufficient to release them alone [22]. To guarantee compliance with ethical, legal and safety-related requirements, various techniques for verification may be used. Verification in general is the confirmation that specified requirements have been met through objective evidence (e.g., inspections, alternative calculations or document reviews) [23].

In their paper, the authors of [22] provide a comprehensive overview of techniques for the formal verification of autonomous robotic systems. Established and strongly focused are heavyweight highly formal techniques such as model checking, integrated formal methods and theorem proving. However, each of these techniques has specific challenges and limitations, such as the disadvantage of complex modeling and limited transferability between programs, the expert knowledge required and the lack of generalizable approaches, which may impair the application and efficiency of these methods. In addition to these techniques, runtime verification is said to be highly efficient, especially on the running system [22].

Runtime verification or runtime monitoring is a lightweight formal method for analyzing just a single execution of a system [24]. It is therefore not reliant on a model or additional obligations to provide evidence of the correctness of the abstraction or model creation [25]. Through thorough monitoring of the system itself and a comparison with requirements, it can perform reliable verification of system executions at runtime.

Runtime verification has proven to be a practical way of detecting and avoiding erroneous or unsafe behavior, which in the context of UAVs may be a violation of minimum separation distances or entering restricted zones [26].

In general, three steps are required to develop a component for runtime verification: specification and formalization of requirements, configuration of a monitoring component and connecting it to the target system [24]. The last two steps mainly deal with inference and information access and end up being highly automated. However, the formulation of requirements is crucial to ensure that the following steps produce correct results. This demands human-provided formalized input.

For the formalization of human knowledge, rules are particularly well suited [27]. Taking advantage of this aspect, the concept of rule-based runtime verification has emerged. The rule-based specifications of requirements used here can be constructed intuitively with syntactically simple if–then conditions and are at the same time very expressive [28]. Rules consist of a premise and a conclusion. A premise (the left-hand side of a rule) can check for the presence or absence of a particular fact, so-called atoms, and the conclusion (the right-hand side of the rule) can add or delete facts or produce error messages. If all atoms of a premise become true, the rule is fired; i.e., the action on the right-hand side is executed [28].

Rules represent a specific kind of knowledge. In order to process this knowledge in the course of runtime verification, it needs to be formalized in a machine-readable form.

2.4. Representation and Processing of Rule Knowledge

The use of ontologies is suitable for the unambiguous definition and application of rule knowledge. An ontology is a knowledge base that models a common understanding of a domain. One of the goals of ontologies is to describe information in a machine-readable and unambiguous way so that computer systems, such as a UAV in this case, have a common understanding and can process this information [29]. In other words, this unambiguous and machine-readable meaning of information enables the combination or integration of heterogeneous systems from different manufacturers, thus enabling interoperability [30].

Unlike conventional data formats used in various computer systems, such as XML or JSON, information in an ontology is given as semantics, i.e., meaning and relations. Semantics are implemented in an ontology through terminological knowledge, which expresses class knowledge. This class knowledge basically applies to the domain and defines classes of resources and relations [31]. For example, the class *UAV*, the class *HumanOperator* and the relation *hasHumanOperator* can be defined to associate a pilot with a drone. The most complex relationships can be expressed in ontologies. In addition to terminological knowledge, an ontology consists of assertional knowledge, which expresses factual knowledge. Factual knowledge represents a specific case within the domain by describing actual individuals and their relationships [31]. For example, the definitions of a UAV, *Hexacopter_1*, as an individual of the class *UAV* and a pilot, *John Doe*, of the class *HumanOperator*. In addition, *John Doe* can be defined as the operator of the UAV *Hexacopter_1* with the defined relationship *hasHumanOperator*.

The class knowledge of ontologies enables a major advantage of ontologies: reasoning. New implicit information is derived from already-specified information, the assertional knowledge [32]. Standard reasoning, e.g., checking class memberships, can be extended with custom rules. Additional knowledge can be inferred using such custom rules [33]. Depending on the domain, different rules can be integrated, e.g., rules for checking the unsafe behavior of autonomous UAVs. The use of rules makes decisions and conclusions more transparent and easier to understand compared with other verification techniques. If unsafe behavior is inferred, it can be reconstructed from the rules and their premises.

Reasoning and such rules are supported by the open-world assumption of ontologies. The open-world assumption implies that a knowledge base is potentially always incomplete. Just because a certain fact is not known does not mean that this fact does not exist or is false [34]. The open-world assumption is transferable to the determination of safe behavior. From the limited number of known unsafe behavior patterns and the incomplete information available about a given situation, it cannot be inferred that the behavior is safe, because important information may be missing. However, it can be determined that certain events represent unsafe behavior. In contrast to the open-world assumption, the closed-world assumption assumes that everything that is not known is wrong [34]. The open-world assumption allows for flexibility in dealing with incomplete or changing information, which is the case in an environment as dynamic as verifying (un)safe behavior of UAVs.

Accordingly, ontologies can also be extended by adding new knowledge, both class and factual knowledge. Ontology design patterns (ODPs) are often used in this context. An ODP represents the vocabulary of a common understanding of a community (e.g., standard or law). Such modular ODPs can be combined by importing the corresponding ODPs into an ontology and aligning them via relations [30]. Such an ontology is called an alignment ontology. Despite the combination, the individual ODPs are managed separately and independently. In principle, ontologies are reusable, since the knowledge about a domain is described independently of a specific task. This reusability is enhanced by the use of ODPs because they are standardized concepts [30].

These ODPs are often used to build concrete domain ontologies. In principle, ontologies can be created at different levels of abstraction. For example, upper ontologies focus on general terms such as object or event and are therefore at a very abstract level, while reference ontologies focus on a specific, still abstract area of knowledge such as medicine or engineering. Domain ontologies are even more specific and relate to a concrete domain, such as autonomous drones, and finally, there are application ontologies, which are the most specific and are developed for concrete applications, such as a mission for a UAV. Concepts of the more abstract levels are often specialized in the more specific levels [35].

The method presented in the following section considers a combination of the elements presented here. It takes into account the lack of predictability of autonomous UAVs in mission execution. Considering the lack of predictability of autonomous UAVs in the execution of missions, it focuses only on current behavior. Knowledge-based reasoning

within an ontology is used to verify whether safety-relevant constraints, represented by rule knowledge, are violated or not.

3. Method for Rule-Based Verification of Unmanned Aerial Vehicles

This section presents the developed method for rule-based verification of autonomous unmanned aerial vehicles by means of a knowledge-based system. It may be used both to identify potential hazards prior to mission execution and to detect unsafe behavior during mission execution. Therefore, first the structure of a suitable knowledge base is presented, followed by the formulation of verification rules. Closing the verification process is explained. The method presented here is not intended to replace a fully fledged risk assessment but can usefully supplement one.

3.1. Selective Acquisition and Alignment of Knowledge

This first step of the method serves to create an initial knowledge base. This serves as the necessary foundation in the format of an ontology on which all subsequent method steps are based, such as the formulation of rules for safety constraints. This ontology may be viewed as a catalog of classes and properties that may exist in a specific domain from the perspective of a person using a specific language to talk about the domain [36]. This initial structure is therefore mainly built from class knowledge.

Since safety constraints for UAVs are not directly dependent on specific goals, they instead depend highly on the given scenario, the environment, or internal factors of the UAV itself [15]. It is therefore necessary to first acquire this domain-specific knowledge and to formalize it in a suitable way, i.e., machine-readable, for the UAV. An important source of knowledge for this process is expertise, which is available mainly from human experts or from the literature. Acquiring this expertise is a complex, labor-intensive process and much more than a simple transfer of knowledge into a machine-readable format [29]. Therefore, the acquisition of knowledge may be carried out successfully in various ways [37]. If the resulting ontology fits the domain perfectly, a suitable domain model can be created simply by filling the ontology with the instances [29].

As this ideal is difficult to achieve in reality through a single monolithic ontology, a modular, reusable approach is focused on here. Therefore, several ODPs are used, combined with an application ontology and a domain ontology within an alignment ontology. This approach was first presented in [21].

In a consecutive method, the authors present a way to formalize an exhaustive knowledge base of safe behavior for unmanned autonomous systems. Therefore, firstly, relevant knowledge sources are identified, relevant statements are extracted, and knowledge elements are formalized and incorporated, creating multiple ODPs. ODPs are considered as the main instruments for this knowledge base. An ODP can, for example, contain a classification scheme for UAVs or define various attributes, e.g., the current flight altitude of a UAV.

The approach of [21] is extended here by the representation of an application ontology. This application ontology is used to map the basic vocabulary regarding a mission for autonomous UAVs. It defines the structure of a mission, consisting of goals, constraints and mission execution. The established constraints of a geofence and a safety landing point (SLP) are already integrated here. A geofence represents the maximum operational area available for a mission. It must not be left. An SLP represents a position where a safe landing can be made, e.g., in the event of a communication breakdown. The use of geofences and SLPs is established for automated UAV missions and may be determined, e.g., with QGroundcontrol. They are therefore also considered a necessary component of missions for autonomous UAVs. A mission contains at least one goal. Additional goals may be linked in parallel or sequentially. Goals, geofences and SLPs are determined by positions, among other things.

The method described in [21] finishes with an alignment step. Equivalences or subclassifications can be conducted here in order to seamlessly link the individual elements of

knowledge. Further ODPs can also be included at a later date, e.g., when facing another mission in a different environment.

Following the alignment process, it is now possible to add factual knowledge. For example, individuals can be created for a mission including goals, geofence, an SLP and the associated UAV. Further factual knowledge is generated automatically, especially at runtime. Figure 1 illustrates a fragment of this knowledge base. It focuses on an individual representing a UAV of the class C3. The UAV is associated with a mission as defined by the application ontology. In addition, the UAV has a specific relation to represent the current flight altitude of 130 m.

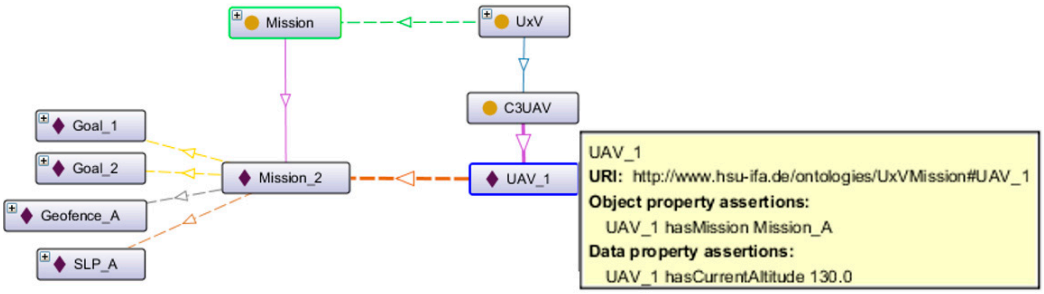


Figure 1. Representation of a specific UAV within the knowledge base. The individual UAV_1 is linked to its class and the individual Mission_A, consisting of a geofence, an SLP and two goals. The UAV also has a current flight altitude.

This process of acquisition and alignment of knowledge creates an exhaustive knowledge base in the format of an ontology containing class knowledge and possibly also factual knowledge. The knowledge base does not yet contain rule knowledge. Thus, for example, the affiliation of a UAV with a certain UAV class can be modeled, and the UAV can be equipped with attributes such as the current flight altitude. However, in order to determine that the current flight altitude represents an exceeded limit and the UAV therefore poses danger, rule knowledge is required.

3.2. Formulation of Rules

This step of the method describes how to systematically formulate rules for safety-related constraints. The success of this step depends largely on expert knowledge of both the constraint and the previously created knowledge base. To provide additional support for conducting this step, five sequential stages for rule formulation are given:

1. Momentary adjustment;
2. Formulation of an error-oriented if not. . ., then rule;
3. Unambiguous assignment to knowledge elements;
4. Reformulation to an if. . ., then rule;
5. Error classification.

In order to explain these five stages, a simple example is given: The European regulation (EU) 2019/945 ([38]) states that the maximum altitude of a C3 UAV is 120 m. This example objectively represents a legally binding constraint concerning the safe behavior of a UAV. In order to monitor this constraint using a component for rule-based runtime verification, it must be reformulated without altering the substantive content. In this context, it is important that resulting rules always be formulated from the perspective of the UAV to be monitored in order to clearly identify the respective unsafe behavior. To avoid uncertainties within the rule, disjunctions (logical OR) are to be avoided. This means that several similar rules may have to be formulated, which, however, always formulate an unambiguous cause of unsafe behavior in the respective premise.

In stage 1 (momentary adjustment), the constraint is initially adjusted so that it can capture the current moment. In order to permanently comply with the constraint, the given limit of a maximum altitude must be maintained at all times. For this purpose, the current

altitude of the UAV must be taken into account and compared with the maximum altitude value. After stage 1 has been applied to the example, an initial rule states: "The current altitude of a C3UAV is not greater than 120 m".

As stated in Section 2.2, it is challenging to ascertain that perfect safety has been achieved. This state is also difficult to harmonize with the open-world assumption of ontologies. Nevertheless, it is feasible to identify individual violations of safety-related constraints representing unsafe behavior. Detecting such violations therefore becomes crucial for enhancing overall safety. Thus, when verifying the behavior of autonomous UAVs, unsafe behavior is of primary interest. This must consequently be detected and reported so that a human operator can initiate a counteraction. Rules must therefore be formulated in an error-oriented manner within stage 2 (formulation of an error-oriented if not. . ., then rule). This is merely a syntactic adaptation. If a given constraint is violated, and thus not fulfilled, an error results. This means that the cause of unsafe behavior, here the exceeding of a limit value, must be formulated in the premise of a rule. It is also important that the premise of a rule always contains an atom representing the specific UAV. Completing stage 2, the rule states: "IF NOT (The current altitude of a C3UAV is not greater than 120 m) THEN Error".

In stage 3 (unambiguous assignment to knowledge elements), all atoms of the premise are to be assigned to existing knowledge elements of the previously created knowledge base. This step is not trivial, as it requires detailed knowledge of the various elements within the ontology and must not violate the meaning of the original constraint. For the given example, the class *C3UAV*, as a subclass of *UAV*, and the property *hasCurrentAltitude* were identified. The modified rule states: "IF NOT (There is a *C3UAV* AND this *C3UAV* has a value *hasCurrentAltitude* AND the value of *hasCurrentAltitude* is not greater than 120 m) THEN Error".

Subsequently, the preceding universal negation of the premise must be resolved within stage 4 (reformulation to an if. . ., then rule). The reformulation of the negation of linked atoms is defined by Boolean algebra. Equation (1) shows that the negation ($\neg$) of a conjunction ($\wedge$) is the disjunction ($\vee$) of negations.

$$\neg(A \wedge B) = (\neg A \wedge B) \vee (A \wedge \neg B) \vee (\neg A \wedge \neg B) \quad (1)$$

The atoms of the premise separated by disjunction represent a logically correct reformulation of the rule. However, the disjunction also means that it is sufficient for a single disjunction to be fulfilled in order to fire the entire rule. Therefore, it is important to identify the parts of the premise that actually cause unsafe behavior and to remove any disjunction that does not contribute to it. In the example, a negation of the classification does not lead to unsafe behavior and is consistently removed. It merely remains: "IF there is a *C3UAV* AND this *C3UAV* has a value *hasCurrentAltitude* AND NOT the value of this *hasCurrentAltitude* is not greater than 120 m THEN Error". (The double negation is to be resolved conclusively but is retained for reasons of presentation.)

Finally, in stage 5 (error classification), the conclusion of the rule must be specified more precisely. The usage of predefined error states in the conclusion facilitates the classification and a subsequent counteraction to those errors. The established terms hazard, danger and harm are used for this purpose. Harm, for example, is defined in ISO 12100 ([39]) and is specified according to the severity of the harm, ranging from minor to critical. This also makes it possible to prioritize the counter-reaction to errors by, e.g., giving preference to a critical harm over an environmental hazard. The error states and their respective subdivisions are also part of the presented application ontology. This results in the final rule: "IF there is a *C3UAV* AND this *C3UAV* has a value *hasCurrentAltitude* AND the value of this *hasCurrentAltitude* is greater than 120 m THEN this *C3UAV* causes a *MinorHarm*" (the classification of this damage as minor is not universally valid and depends on the respective risk assessment). At this point, it can be seen that the conclusion also serves to link the determined error state with the UAV atom of the premise.

There are different rule languages for creating rule knowledge in ontologies, some of which can be converted into one another [40]. In the remainder of this paper, the lightweight and established rule language SWRL ([41]) is used. In Table 1, the final example rule is displayed as an SWRL-Rule. The premise of the rule is created from atoms for the classification of the UAV (line 1), the query of the current altitude of this UAV (line 2) and the comparison of this altitude with the maximum value of 120 m (line 3), each linked via conjunctions (circumflex symbol). The conclusion in line 4 is indexed with an arrow symbol. It links the UAV with the defined error state. The addition of an explanatory text within the conclusion is not necessary but may support a human operator understanding the respective cause of unsafe behavior.

Table 1. SWRL-Rule for unsafe behavior caused by exceeding the maximum altitude.

Line No.	SWRL-Atom
1	*C3UAV(?myUAV)*
2	*^ hasCurrentAltitude(?myUAV, ?altitude)*
3	*^ swrlb:greaterThan(?altitude, 120.0, "xsd:float")*
4	*−> MinorHarm(?myUAV, "The UAV has exceeded the maximum altitude")*

In SWRL, variables are indexed with question marks. Instead of these variables, it is also possible to refer to concrete individuals of the ontology. For example, exception rules that do not apply to all instances of a class can be created in this way.

Some rules may already be checked before the start of the mission. They are used to detect hazardous missions. Those missions cannot be executed without violating constraints. Thus, UAVs tasked with such missions pose potential hazards.

3.3. Processing of Verification before and during a Mission

To perform the process of runtime verification, two steps must be repeated alternately and permanently: updating factual knowledge and inferring it. At runtime, the UAV therefore adds new factual knowledge and modifies existing factual knowledge within the ontology. Each observation and each action of the UAV is formulated as precisely as possible with individuals and properties according to the created template of class knowledge. For example, a relation to the current flight altitude is linked to the corresponding individual representing a UAV. The current altitude must be continuously modified here as a literal. After each update of the factual knowledge, an inference process is conducted. This process is mainly concerned with working through all existing rules. It checks whether the individual atoms of a particular premise apply. If all partial elements of the premise of a rule are fulfilled, it fires, and the condition is executed. In the context of this method, an error state is therefore attached to the individual of the UAV concerned. Any new error must be immediately communicated to the human operator.

In order to exploit the full potential of the method, it is evident that the UAV must provide access to information about its actions and observations. The update rate of this information is highly relevant. There are various so-called reasoners for automated inference within an ontology, many of which are suitable for inferring from rule knowledge.

In the next section, the method described here is illustrated by means of a practical use case. In order to enable a comparison of the respective method steps, the development of the knowledge base, the formulation of rules and the processing of verification are also separated into subsections.

4. Implementation and Use Case

This section demonstrates the implementation of a rule-based verification component based on the method presented in Section 3. Therefore, a use case for traffic monitoring is considered. The use case is described both with missions in a simulation-based environment and a real-life environment. The implementation was conducted using the

system architecture presented by the authors of [42]. Figure 2 provides an overview of this system architecture.

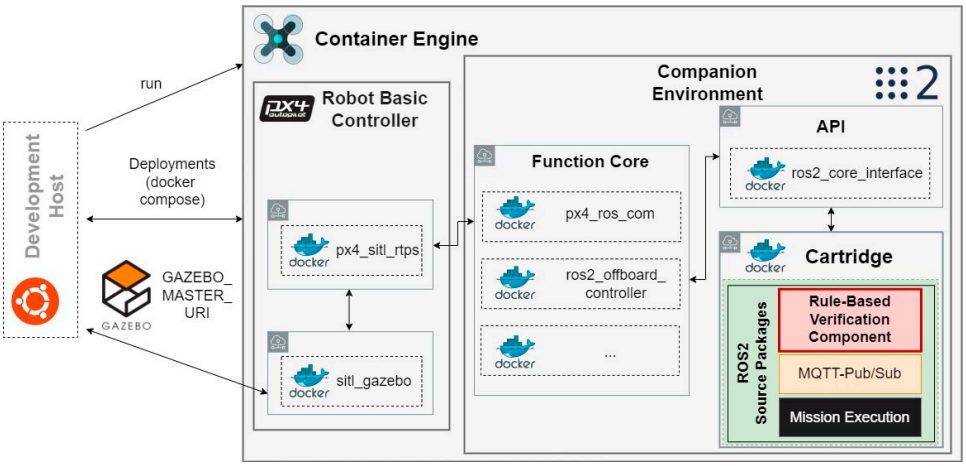


Figure 2. Overview on the used system architecture (adapted from [42]).

The system architecture outlined in [42] was developed to deploy heterogenous autonomous mobile robots in both simulated and real-world scenarios and is primarily based on the Robot Operation System 2 (ROS2) [43]. To simplify the integration of the various software components used, the container technology Docker is used, which additionally guarantees independence from the respective host system. Gazebo, a physis simulator, is chosen to emulate the real world and simulate robot behavior, providing sensor data. The separation of server and client reduces computing effort, with the Gazebo client on the host system and the Gazebo server in a Docker container. Each robot features a specific Container Engine containing a Robot Basic Controller and a Companion Environment. While [42] addresses various heterogeneous robots, this paper focuses solely on aerial robots, i.e., UAVs.

The Robot Basic Controller, represented here by the px4 platform, manages control mechanisms, such as UAV rotor control. The associated px4 autopilot software (here the v1.14 release is used) integrates with Gazebo for command transmission and simulation data reception. The Companion Environment comprises the Function Core, overseeing functional processes at a higher abstraction level than the Robot Basic Controller. It is linked to the Robot Basic Controller via a Real-Time Publish Subscribe bridge. This unit encapsulates basic functions, such as the movement to given waypoints, offering a single peripheral interface for execution within a ROS2 environment.

The Cartridge within the Companion Environment serves as an interface for developers, facilitating individual development, testing and code variations. Within the Cartridge, the containerized ROS2 packages for the Rule-Based Verification Component (highlighted in red) and a communication module (highlighted in orange) were integrated. The capabilities for mission execution of the respective UAV are considered a black box within the Cartridge (highlighted in black) and are not further described, since the focus is on explaining the function of the Rule-Based Verification Component. Development and simulation-based testing of further Cartridge elements can be conducted easily via an Ubuntu-based host system.

For real-life UAVs, the px4 autopilot software operates on an associated hardware controller. The Companion Environment can be similarly implemented on the actual UAV using a separate high-performance companion computer, such as a maker board. The containerized elements of the Cartridge can thus be transferred to the real UAV without modification. The platform approach presented by these authors makes it possible to easily develop and implement so-called creative functions for UAVs without having to exercise direct control over the actuators used; thus, no detailed expertise on the UAVs architecture is needed.

In the following, first the use case is described, and then the considered knowledge base is presented. After this, some exemplary verification rules are formulated and explained, and their processing is shown for selected events.

4.1. Description of the Considered Use Case

In their paper, the authors of [44] examine research in the area of UAVs used for traffic monitoring. They emphasize the high potential of UAVs for traffic monitoring. However, the implementation of real-life scenarios is said to be particularly hindered due to regulatory constraints. As the primary legal concern revolves around regulatory matters, given that aviation regulations are primarily designed to ensure an appropriate standard of safety ([2]), the use case presented here focuses on two missions for UAVs for traffic monitoring in consideration of safety-relevant regulatory constraints. It is not within the scope of this paper to present new approaches for the processing or analysis of traffic data. The actual process of traffic monitoring at the selected locations therefore is of secondary importance in the following.

A simulation-based mission using the physis simulator Gazebo and a simplified real-life mission are presented. The use case is intended to show the treatment of constraints considered with both external and internal dependencies. For this purpose, a special focus is given to constraints due to geographical zones and constraints due to UAV classification. In mission M1, a quadcopter UAV is used in a simulation-based environment. The quadcopter is a class C5 UAV of the Specific category. The mission M1 is located in a part of the German city of Hamburg. The real-life mission M2 uses a commercial hexacopter. The hexacopter is a class C3 UAV of the Open category. Mission M2 is located on a scaled-down test field. Within their missions, the UAVs are tasked with monitoring traffic at multiple selected locations (goals) within a given road network. In order to enable the maximum range of actions, no specifications are made regarding the process of goal achievement. Thus, no predetermined sequence of the mission-goals is made. Figure 3 shows a top view of the two missions and respective goals for the simulated mission M1 (a) and the real-life mission M2 (b). Each mission contains five goals. Their numbering does not represent a sequence. Each UAV is located near goal no. 1 at the start of the mission. Geographical zones (colored polygons) and the mission-specific geofence (red polygon) are also shown here. Due to its given nature, the test field unfortunately does not contain a visible road network. As the test field does not have the same geographical dimensions as the simulation environment, geographical zones, maximum velocity and maximum altitude had to be scaled down.

(**a**) (**b**)

Figure 3. Top view of the presented missions showing the position of the UAV, goal locations, the environment, geographical zones and the geofence for the simulation-based mission M1 (**a**) and the real-life mission M2 (**b**).

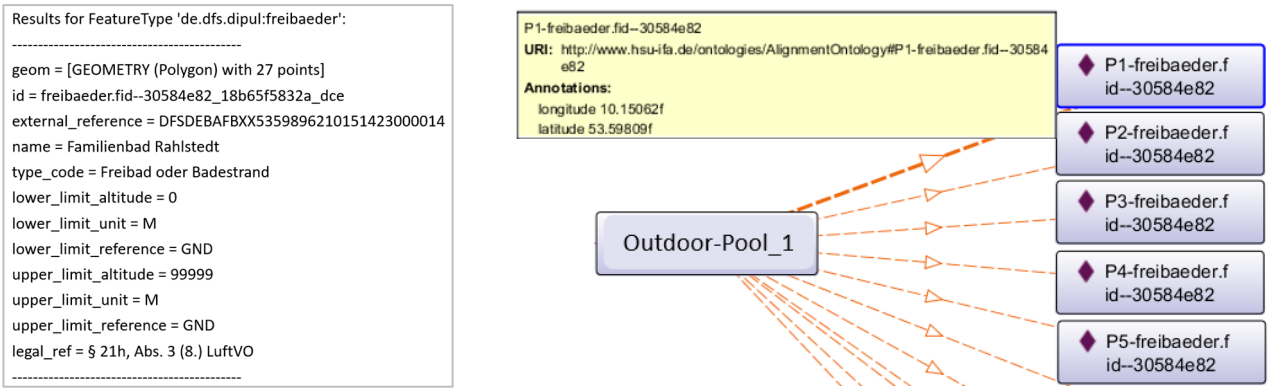
In order to execute its respective mission, the UAV has to fly to every goal and monitor the traffic at these locations. Any possible violation of constraints has to be detected by the rule-based verification component and reported to a human operator. This component is designed as an add-on and thus cannot influence the behavior of the UAV itself. Instead, the human operator is provided three limited means of control: abort the mission by returning and landing on the SLP, abort the mission by immediate landing or ignore the report and continue the mission.

4.2. Considered Knowledge Base

In order to conduct safe traffic monitoring missions with UAVs, it must be ensured that no constraints are violated. This applies in particular to safety-related legal constraints. Various legal regulations impose requirements on the operation of UAVs. For example, the European regulations (EU) 2019/947 ([13]), (EU) 2019/945 ([38]) and (EU) 2020/1058 [45], as well as the German national regulation Luftverkehrsordnung ([46]), are relevant for these missions, which are situated in a major German city. Together, these three European regulations result in a comprehensive classification and categorization for UAVs. Depending on the class and category, different velocities or maximum takeoff masses apply. These specifications apply throughout the entire scope of the regulations. In simple terms, they therefore only depend on the internal dependencies of the UAV (as long as the UAV does not leave the borders of the EU). The regulation [46] is a national supplement to the Standardized European Rules of the Air. Among other things, it describes the type and scope of geographical zones and the conditions under which UAVs may operate inside them. These geographical zones only exist at certain locations and therefore represent constraints with mainly external dependencies.

In the first step of the method, the class knowledge of the EU regulations and the national regulation will first be assembled and aligned. This serves to create a machine-readable knowledge base as a foundation for the following steps. In [21], a method was presented with which ODPs can be created and aligned from legal regulations. This was used to create an ODP for the regulation [46]. Further ODPs for the regulations [13,38] have already been published by the authors of [47] and can therefore be easily imported here. Regulation [45], which is a supplement to [38], was again created using the method from [21]. These four ODPs are supplemented by the application ontology described in Section 3.1 and combined into an alignment ontology. This alignment ontology now represents an exhaustive and detailed knowledge base for the considered use case. It is available online at *github.com/RIVA_Safety*. It is possible to integrate additional ODPs into this alignment ontology to map further use cases. To improve the illustration of the application, the repository also contains a video of a simulation-based use case.

In the following, mission-specific knowledge is integrated into the form of factual knowledge. First, the necessary individuals are created for the missions, goals, geofence and SLP. Next, individuals are automatically created for geographical zones within or intersecting the geofence. To this end, more than 30 European countries provide different platforms that define such zones. For Germany, this is the platform dipul ([48]) provided by the Federal Ministry for Digital and Transport. Using the associated application programming interface (API), it is possible to automatically retrieve information about the zones. Figure 4 shows the information provided by the API for an outdoor pool within the area of mission M1 (a) and its transfer to the ontology as an individual with multiple corner points (b). For reasons of simplification, this individual is referred to as "Outdoor-Pool_1" in the following. In addition to a classification of the zone, the API provides a unique identifier as well as lower and upper limits. The coordinates of the corner points can also be determined using the API. It should be noted that the spatial extent of a zone within the ontology only takes into account the intersection with the mission-specific geofence. Other geographical zones or parts thereof that are located outside the geofence are not included in the ontology.



(**a**) (**b**)

Figure 4. Information regarding a specific geographical zone provided by the platform dipul ([48]) (**a**) and representation within the ontology (**b**).

For the smaller real-life mission M2, similar geographical zones had to be created manually instead. The triangular orange zone shown in Figure 3b is also intended to represent an outdoor pool. The corresponding individual in the respective ontology is given the name "Outdoor-Pool_2" and three corner points.

Enriched by factual knowledge, it is now possible to formulate both generally valid rules based exclusively on class knowledge and specific rules that also take factual knowledge into account.

4.3. Formulation of Rules

In the following, some rules are exemplarily presented to express constraints for the two missions. First, the example rule described in Table 1 is also used here, but in a scaled version with a maximum altitude limit of 12 m. Second, the geographical zone from Figure 3 is used to demonstrate how individuals are taken into account in rules.

The regulation [46] permits the operation of UAVs above outdoor swimming pools outside opening hours. Consequently, when conducting stage 1 (momentary adjustment) of rule formulation, the resulting initial rule must take into account a combination of the current location of the UAV and the current time. In stage 3 (unambiguous assignment to knowledge elements), in addition to the general applicable classes and properties, the specific individual *Outdoor-Pool_1* needs to be selected. Lastly, the error classification *MediumDanger* is selected within stage 5 (error classification). Table 2 shows the resulting SWRL-Rule. It applies to any UAV, or subclass thereof, in relation to the Outdoor-Pool_1 referred to. While the respective locations are compared in line 2, the time check is carried out in lines 3–5.

Table 2. SWRL-Rule for unsafe behavior concerning a specific geographical zone.

Line No.	SWRL-Atom
1	*UAV(?myUAV)*
2	*^ isAbove(?myUAV, Outdoor-Pool_1)*
3	*^ hasTimeStamp(?myUAV, ?currentTime)*
4	*^ swrlb:greaterThanOrEqual(?currentTime, "09:00:00", "xsd:dateTime")*
5	*^ swrlb:lessThanOrEqual(?currentTime, "17:00:00", "xsd:dateTime")*
6	*−> MediumDanger(?myUAV, "The UAV operates above Outdoor-Pool_1 within opening hours")*

An equivalent rule is created for Outdoor-Pool_2. When compared with Table 1, the modularity of the rules is highlighted here. The complexity of rules can be increased by adding any number of atoms.

Other rules relevant to the use case describe the minimum altitude over residential areas of 100 m and the restriction on flying over the hatched zones shown in Figure 3.

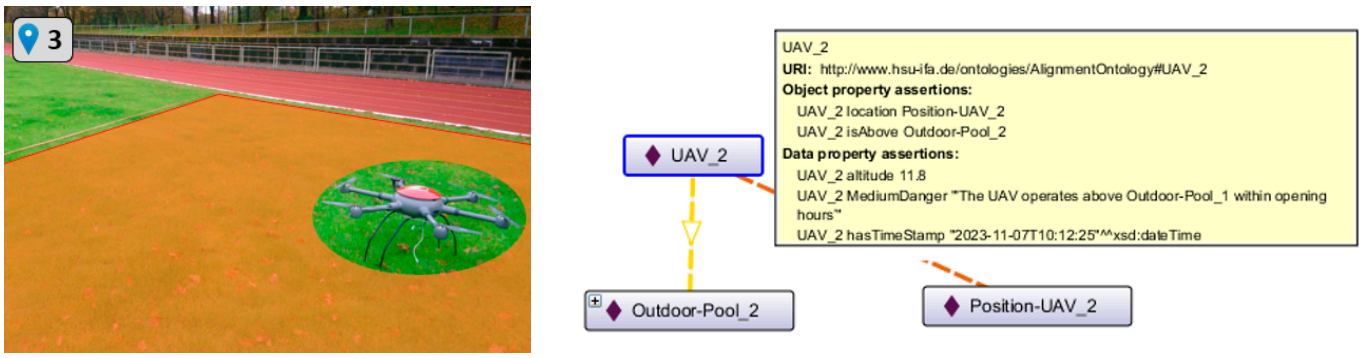
As soon as this rule knowledge has been added to the ontology, this can be transmitted to the respective UAV, where it is used for runtime verification.

4.4. Use Case Execution and Processing of Rules

The execution of the use case is described using selected events from the two missions. As soon as the ontology is transmitted to the respective UAV, the first task of the verification component is to check the respective given mission. Herewith, no hazards are identified for mission M1 before execution. In the case of mission M2, goal no. 4 cannot be reached without violating constraints. Either the hatched geographical zone would have to be flown over without permission or the geofence would have to be left. The verification component therefore reports a hazard. Consequently, mission M2 is adjusted so that goal no. 4 is removed from it.

In mission M1, traffic monitoring is carried out at the individual goals at a height of 80 m. Between the goals, the UAV flies at an altitude of 120 m. In some cases, the UAV is still climbing while flying over residential areas and has an actual flight altitude of less than 100 m. The verification component reports the resulting danger. However, the human operator decides to ignore the report and continue the mission. Mission M1 is therefore successfully completed.

In mission M2, traffic monitoring is carried out at the individual goals at a height of 8 m. Between the goals, the UAV flies at an altitude of 12 m. While the UAV moves from goal no. 3 towards goal no. 2, it does not avoid the zone of Outdoor_Pool-2. When the UAV enters the zone at 10 a.m., the associated rule fires and the human operator is warned. He decides to abort the mission with an immediate landing. Figure 5 shows the landed UAV in the geographical zone and the cause of the warning within the ontology.



(**a**) (**b**)

Figure 5. UAV landed inside a geographical zone (zone is highlighted for better visualization) (**a**) and representation of the causing event within the ontology (**b**).

During the time period in which the human operator reacted to the warning and took the decision to land, the UAV kept flying unhindered and moved further into the zone. The landing ends the mission M2. It therefore concludes unsuccessfully.

5. Discussion

In this section, this paper's results will be discussed. The authors emphasize that the presented approach does not aim to outperform other existing approaches in terms of processing time. Given its knowledge-based nature, extensive data processing is necessary, leading to a potential performance drawback. Instead, the focus is on introducing a powerful and broadly applicable approach that effectively addresses various external and internal safety constraints.

Below, firstly some limitations of the presented method and its implementation and possible starting points for further research and investigations are addressed and secondly the necessity of predictive runtime verification is discussed.

5.1. Limitations

The method and implementation presented here should not be regarded as concluded in their entirety, nor free of limitations. The realization of a general interface between the UAV and the runtime verification component appears to be the most important. Although the component is able to process the provided information, it requires explicit knowledge of where the UAV provides which information in which format.

Although the information on geographical zones from the official platform ([48]) is preferable to that provided by other available platforms, it is also incomplete or outdated in parts. A useful addition to this platform is the consideration of perception through sensor technology and its integration and updating within the knowledge base.

In addition, no special attention was paid in this paper to an overflowing knowledge base. In principle, the more knowledge is available, the more can be inferred from it. However, especially for systems with low computing power, too much unnecessary knowledge also increases the effort and duration of the inference, so removing unnecessary ODPs before a mission or the selective deactivation of nonapplicable rules should be considered further. As such, ref. [49] impressively demonstrates the correlation between the size of an ontology and the computing time for different reasoners and thus reinforces the need to keep the respective knowledge base small and simple.

Using the method not only to passively detect unsafe behavior but also to actively counteract it automatically is a promising extension. However, this requires in-depth knowledge of the capabilities and skills of the UAV.

Expanding on the rule-based approach, an implementation with behavior trees is also conceivable in order to enable both an increase in performance and means of counteraction. Behavior trees are particularly suitable for sequential condition chains [50]; thus, they can select an appropriate branch for a UAV counteraction depending on the detected situation [51]. However, they are also generally suitable for the mere detection of unsafe behavior. Transferring the rules determined here into a behavior tree may provide advantages in the performance of the verification component. However, this requires a preliminary corresponding compilation that transfers these rules into branches and leaves of a behavior tree. If the rules within this process are strictly separated, the tree will contain parallels and redundancies, but if the rules are intertwined, the behavior tree will be difficult to maintain.

5.2. Predictive Runtime Verification

The rule-based approach to the verification of autonomous and automated UAVs presented in this paper can be used as a valuable complement to support a human operator monitoring one or multiple UAVs. Due to its modular and expandable design, it is advantageous to use existing geo-awareness functions, as it can also check the conditions of operation for geographical zones. However, this method still only detects states in which unsafe behavior in the form of a hazard, danger or harm has already occurred. In addition, the comparatively complex knowledge processing, especially with low-performance systems, can have an additional negative effect, e.g., where a UAV has already advanced further into a geographical zone before the message is sent. The human reaction time in connection with the transmission times of the warning and the counter-reaction increases this negative effect. The consequence is a temporary unhindered continuation of unsafe behavior, which is illustrated in Figure 5a.

It is therefore worth discussing whether it is sensible to extend the presented method with a predictive component that conducts a warning, e.g., before entering a geographic zone. This predictive component should be available for all existing ontologies and should therefore be able to be integrated into the application ontology.

This means determining the minimum distance at which a UAV comes to a stop in front of a geographical zone or avoids it. If the distance between the UAV and the geographical zone equals this minimum distance, a warning must be transmitted to the human operator.

To initiate a discussion, a concept for determining this minimum warning distance is presented here. The minimum warning distance at which the UAV comes to a standstill in front of the zone depends largely on the reaction time of the human operator and the deceleration of the UAV (assuming delay-free transmission of messages and data processing of the UAV). The reaction distance depends on the reaction time of the human and the current velocity of the UAV. At constant deceleration, the deceleration distance depends on the maximum counterthrust and the current velocity. If the distance between the UAV at Position P1 and a geographical zone is equal to the minimum warning distance, the UAV is stopped on the edge of the zone. Therefore, an additional fixed buffer distance is taken into account, which causes the UAV to be stop at position P2 before the zone. Figure 6 provides a graphical representation of this concept.

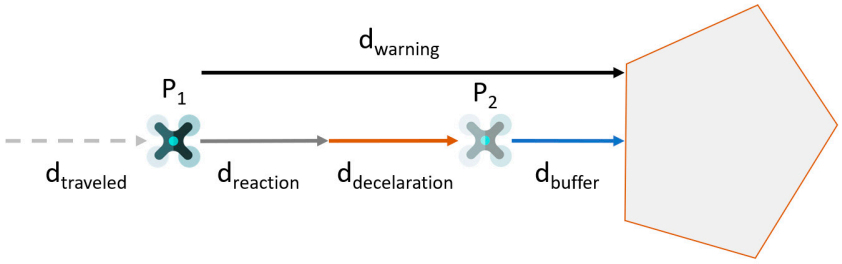


Figure 6. Concept of predictive monitoring. The minimal warning distance equals the sum of the distances of a reaction, deceleration and buffer.

In their paper, the authors of [52] quantified the reaction time of air traffic controllers as 2–3 s. As the literature does not provide any comparable values for human operators of UAVs, this reaction time is considered as an adequate reference value.

6. Conclusions and Outlook

Motivated by the ambivalence about increasing autonomy on the one hand and the demand for safety and controllability on the other, an approach to detecting unsafe behavior of autonomous UAVs was presented here. A method for rule-based runtime verification was developed for this purpose. By using a knowledge-based system in the format of an ontology, it is possible to formalize rule knowledge and draw conclusions from it with the help of inference. For this purpose, factual knowledge is automatically updated and generated within the ontology. The method is designed to be modular and repeatable and can therefore be expanded to include further domain knowledge for different scenarios and applications. A simple use case was implemented to demonstrate the functionality of the method, and the knowledge base developed for this is also openly available. Furthermore, limitations of the presented approach, especially with regard to predictivity, were discussed. Some of these current limitations offer potential for interesting future research.

To conclude, the approach presented for rule-based verification was demonstrated to be effective. Using a knowledge-based system instead of data-driven approaches only requires little expertise about the respective UAV, which enables reusability and adaptability to multiple different UAVs. Since this approach is designed as a monitoring add-on that does not conduct automated counteractions, it is also possible to combine it with existing safety components without causing conflicts.

However, it should also be noted that the approach in its current form has potential to increase its efficiency. In addition to implementation-dependent factors such as the computing power of the respective UAV, method-dependent factors, such as the size of the respective knowledge base, should primarily be taken into account.

It is to be expected that within the still-developing field of UAVs, further legal, normative and technical constraints for the safe operation of UAVs will be formulated, and existing constraints may be amended. The resulting need for a generalizable and maintainable approach can therefore be solved by the rule-based approach presented here. A rule base created in this way may even be certified by authorities.

The authors intend to focus further research on the interface between the autonomous UAV and the component for rule-based runtime verification. Concepts of communication for software agents offer potential for successful interoperability. Furthermore, it is intended to extend the presented approach to other modalities and to enable the monitoring of a team of heterogeneous autonomous robots that must comply with different rules.

Author Contributions: Conceptualization, C.S.; methodology, C.S., L.M.V.d.S. and K.G.; software, C.S.; validation, C.S. and A.F.; writing—original draft preparation, C.S.; writing—review and editing, C.S., L.M.V.d.S. and A.F.; visualization, C.S.; supervision, A.F.; project administration, A.F.; funding acquisition, A.F. All authors have read and agreed to the published version of the manuscript.

Funding: This research is funded by dtec.bw—Digitalization and Technology Research Center of the Bundeswehr which we gratefully acknowledge (project RIVA). dtec.bw is funded by the European Union—NextGenerationEU.

Data Availability Statement: The ontologies presented in the context of this paper are openly available on github.com/RIVA_Safety (accessed on 2 January 2024).

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Ul Husnain, A.; Mokhtar, N.; Mohamed Shah, N.; Dahari, M.; Iwahashi, M. A Systematic Literature Review (SLR) on Autonomous Path Planning of Unmanned Aerial Vehicles. *Drones* **2023**, *7*, 118. [CrossRef]
2. Konert, A.; Dunin, T. A Harmonized European Drone Market?—New EU Rules on Unmanned Aircraft Systems. *Adv. Sci. Technol. Eng. Syst. J.* **2020**, *5.3*, 93–99. [CrossRef]
3. Cummings, M.L. Operator Interaction with Centralized Versus Decentralized UAV Architectures. In *Handbook of Unmanned Aerial Vehicles*; Valavanis, K.P., Vachtsevanos, G.J., Eds.; Springer: Dordrecht, The Netherlands, 2015; pp. 977–992, ISBN 9789048197064.
4. Floreano, D.; Wood, R.J. Science, technology and the future of small autonomous drones. *Nature* **2015**, *521*, 460–466. [CrossRef] [PubMed]
5. Rasmussen, S.; Kingston, D.; Humphrey, L. A Brief Introduction to Unmanned Systems Autonomy Services (UxAS). In Proceedings of the 2018 International Conference on Unmanned Aircraft Systems, ICUAS' 18, Dallas, TX, USA, 12–15 June 2018; IEEE: Piscataway, NJ, USA, 2018; pp. 257–268, ISBN 978-1-5386-1354-2.
6. Unmanned Aircraft Systems and the Concepts of Automation and Autonomy. Available online: https://www.eurocockpit.be/positions-publications/unmanned-aircraft-systems-and-concepts-automation-and-autonomy (accessed on 24 October 2023).
7. Chen, Y.; Zhang, G.; Zhuang, Y.; Hu, H. Autonomous Flight Control for Multi-Rotor UAVs Flying at Low Altitude. *IEEE Access* **2019**, *7*, 42614–42625. [CrossRef]
8. Joyo, M.K.; Hazry, D.; Faiz Ahmed, S.; Tanveer, M.H.; Warsi, F.A.; Hussain, A.T. Altitude and horizontal motion control of quadrotor UAV in the presence of air turbulence. In Proceedings of the 2013 IEEE Conference on Systems, Process & Control (ICSPC 2013), Kuala Lumpur, Malaysia, 13–15 December 2013; pp. 16–20. [CrossRef]
9. DJI FlySafe. Available online: https://fly-safe.dji.com/home (accessed on 31 December 2023).
10. Huang, H.M.; Pavek, K.; Novak, B.; Albus, J.; Messin, E. A framework for autonomy levels for unmanned systems (ALFUS). In Proceedings of the AUVSI's Unmanned Systems North America, Baltimore, MD, USA, 28–30 June 2005; pp. 849–863.
11. Parasuraman, R.; Sheridan, T.B.; Wickens, C.D. A model for types and levels of human interaction with automation. *IEEE Trans. Syst. Man. Cybern. A Syst. Hum.* **2000**, *30*, 286–297. [CrossRef] [PubMed]
12. *Regulation (EU) 2018/1139 of the European Parliament and of the Council of 4 July 2018 on Common Rules in the Field of Civil Aviation and Establishing a European Union Aviation Safety Agency, and Amending Regulations (EC) No 2111/2005, (EC) No 1008/2008, (EU) No 996/2010, (EU) No 376/2014 and Directives 2014/30/EU and 2014/53/EU of the European Parliament and of the Council, and Repealing Regulations (EC) No 552/2004 and (EC) No 216/2008 of the European Parliament and of the Council and Council Regulation (EEC) No 3922/91: (EU) 2018/1139*; Publications Office of the European Union: Luxembourg, 2018.
13. *Commission Implementing Regulation (EU) 2019/947 of 24 May 2019 on the Rules and Procedures for the Operation of Unmanned Aircraft: (EU) 2019/947*; Publications Office of the European Union: Luxembourg, 2019.
14. Worpenberg, C. Level 3, 4 oder 5? "Autonome" Fahrzeuge zu Land, Wasser und in der Luft—Ein Plädoyer für eine verkehrsbereichsübergreifende Begriffseinführung von Automatisierungs- und Autonomiegraden. *InTeR Z. Zum Innov. Tech.* **2022**, *3*, 98–106.

15. Sieber, C.; Vieira da Silva, L.M.; Fay, A. Agilität durch Auflagen—Unterstützung der Missionsplanung für Autonome Roboter. In *Automation 2023*; VDI Verlag: Düsseldorf, Germany, 2023; pp. 691–704, ISBN 9783181024195.
16. PX4 Autopilot. Open Source Autopilot for Drones—PX4 Autopilot. Available online: https://px4.io/ (accessed on 2 November 2023).
17. QGroundControl—Drone Control. QGC—QGroundControl—Drone Control. Available online: http://qgroundcontrol.com/ (accessed on 2 November 2023).
18. Ramirez-Atencia, C.; Camacho, D. Extending QGroundControl for Automated Mission Planning of UAVs. *Sensors* **2018**, *18*, 2339. [CrossRef] [PubMed]
19. ISO/IEC GUIDE 51:2014—IEC-Normen—VDE VERLAG. Available online: https://www.vde-verlag.de/iec-normen/220702/iso-iec-guide-51-2014.html (accessed on 23 November 2022).
20. Schnieder, E.; Schnieder, L. *Verkehrssicherheit: Maße und Modelle, Methoden und Maßnahmen für den Straßen- und Schienenverkehr*; Springer Vieweg: Berlin, Germany, 2013; ISBN 978-3-540-71032-5.
21. Sieber, C.; Worpenberg, C.; Vieira da Silva, L.M.; Schuler-Harms, M.; Fay, A. Acquisition and Formalization of Knowledge to Ensure Safe Behavior of Heterogenous Unmanned Autonomous Systems—An Interdisciplinary Approach. In Proceedings of the 2023 International Conference on Unmanned Aircraft Systems (ICUAS), Warsaw, Poland, 6–9 June 2023; IEEE: Piscataway, NJ, USA, 2023; pp. 199–206, ISBN 979-8-3503-1037-5.
22. Luckcuck, M.; Farrell, M.; Dennis, L.A.; Dixon, C.; Fisher, M. Formal Specification and Verification of Autonomous Robotic Systems. *ACM Comput. Surv.* **2020**, *52*, 1–41. [CrossRef]
23. *EN ISO 9000:2015-11*; Quality Management Systems—Fundamentals and Vocabulary (ISO 9000:2015); German and English version EN ISO 9000:2015. Beuth Verlag GmbH: Berlin, Germany, 2015.
24. Bartocci, E.; Falcone, Y.; Francalanza, A.; Reger, G. Introduction to Runtime Verification. In *Lectures on Runtime Verification: Introductory and Advanced Topics*; Bartocci, E., Falcone, Y., Eds.; Springer: Cham, Switzerland, 2018; pp. 1–33, ISBN 978-3-319-75631-8.
25. Sokolsky, O.; Roşu, G. Introduction to the special issue on runtime verification. *Form. Methods Syst. Des.* **2012**, *41*, 233–235. [CrossRef]
26. Vierhauser, M.; Cleland-Huang, J.; Bayley, S.; Krismayer, T.; Rabiser, R.; Grunbacher, P. Monitoring CPS at Runtime—A Case Study in the UAV Domain. In Proceedings of the 2018 44th Euromicro Conference on Software Engineering and Advanced Applications (SEAA), Prague, Czech Republic, 29–31 August 2018; IEEE: Piscataway, NJ, USA, 2018.
27. Hayes-Roth, F. Rule-based systems. *Commun. ACM* **1985**, *28*, 921–932. [CrossRef]
28. Havelund, K. Rule-based runtime verification revisited. *Int. J. Softw. Tools Technol. Transf.* **2015**, *17*, 143–170. [CrossRef]
29. Studer, R.; Benjamins, V.; Fensel, D. Knowledge engineering: Principles and methods. *Data Knowl. Eng.* **1998**, *25*, 161–197. [CrossRef]
30. Hildebrandt, C.; Kocher, A.; Kustner, C.; Lopez-Enriquez, C.-M.; Muller, A.W.; Caesar, B.; Gundlach, C.S.; Fay, A. Ontology Building for Cyber–Physical Systems: Application in the Manufacturing Domain. *IEEE Trans. Automat. Sci. Eng.* **2020**, *17*, 1266–1282. [CrossRef]
31. Hitzler, P.; Krötzsch, M.; Rudolph, S. *Foundations of Semantic Web Technologies*; Chapman & Hall: Boca Raton, FL, USA, 2009; ISBN 978-1-4200-9050-5.
32. Grimm, S.; Abecker, A.; Völker, J.; Studer, R. Ontologies and the Semantic Web. In *Handbook of Semantic Web Technologies*; Domingue, J., Fensel, D., Hendler, J.A., Eds.; Springer: Berlin/Heidelberg, Germany, 2011; pp. 507–579, ISBN 978-3-540-92912-3.
33. Hitzler, P.; Parsia, B. Ontologies and Rules. In *Handbook on Ontologies*; Staab, S., Studer, R., Eds.; Springer: Berlin/Heidelberg, Germany, 2009; pp. 111–132, ISBN 978-3-540-70999-2.
34. Antoniou, G.; van Harmelen, F. Web Ontology Language: OWL. In *Handbook on Ontologies*; Staab, S., Studer, R., Eds.; Springer: Berlin/Heidelberg, Germany, 2009; pp. 91–110, ISBN 978-3-540-70999-2.
35. Goncalves, P.J.; Olivares-Alarcos, A.; Bermejo-Alonso, J.; Borgo, S.; Diab, M.; Habib, M.; Nakawala, H.; Ragavan, S.V.; Sanz, R.; Tosello, E.; et al. IEEE Standard for Autonomous Robotics Ontology [Standards]. *IEEE Robot. Automat. Mag.* **2021**, *28*, 171–173. [CrossRef]
36. Lebbink, H.J.; Witteman, C.L.; Meyer, J.J.C. Ontology-based knowledge acquisition for knowledge systems. In Proceedings of the 14th Dutch-Belgian Artificial Intelligence Conference (BNAIC'02), Leuven, Belgium, 21–22 October 2002; pp. 195–202.
37. Wielinga, B.J.; Bredeweg, B.; Breuker, J.A. Knowledge acquisition for expert systems. In *Advanced topics in artificial intelligence: 2nd Advanced Course, ACAI '87, Oslo, Norway, July 28–August 7, 1987*; Nossum, R.T., Nossum, R., Eds.; Springer: Berlin/Heidelberg, Germany, 1988; pp. 96–124, ISBN 9783540506768.
38. *Commission Delegated Regulation (EU) 2019/945 of 12 March 2019 on Unmanned Aircraft Systems and on Third-Country Operators of Unmanned Aircraft Systems: (EU) 2019/945*; Publications Office of the European Union: Luxembourg, 2019.
39. *ISO 12100:2010(en)*; Safety of Machinery—General Principles for Design—Risk Assessment and Risk Reduction, 13.110 (12100). ISO: Geneva, Switzerland, 2010.
40. Bassiliades, N. A Tool for Transforming Semantic Web Rule Language to SPARQL Infererecing Notation. *Int. J. Semantic Web Inf. Syst.* **2020**, *16*, 87–115. [CrossRef]
41. SWRL: A Semantic Web Rule Language Combining OWL and RuleML. Available online: https://www.w3.org/submissions/SWRL/ (accessed on 20 November 2023).

42. Sieber, C.; Vieira da Silva, L.M.; Fay, A.; Brogt, T.; Strobel, G.; Berkowitz, S.; Zembrot, L. *A Universal Approach to Command and Control Heterogeneous Autonomous Robots*; dtec.bw-Beiträge der Helmut-Schmidt-Universität/Universität der Bundeswehr Hamburg: Forschungsaktivitäten im Zentrum für Digitalisierungs- und Technologieforschung der Bundeswehr dtec.bw; Helmut-Schmidt-Universität: Hamburg, Germany, 2022; pp. 276–280. [CrossRef]
43. Macenski, S.; Foote, T.; Gerkey, B.; Lalancette, C.; Woodall, W. Robot Operating System 2: Design, architecture, and uses in the wild. *Sci. Robot.* **2022**, *7*, eabm6074. [CrossRef] [PubMed]
44. Kanistras, K.; Martins, G.; Rutherford, M.J.; Valavanis, K.P. A survey of unmanned aerial vehicles (UAVs) for traffic monitoring. In Proceedings of the 2013 International Conference on Unmanned Aircraft Systems (ICUAS 2013), Atlanta, GA, USA, 28–31 May 2013; IEEE: Piscataway, NJ, 2013; pp. 221–234, ISBN 978-1-4799-0817-2.
45. *Commission Delegated Regulation (EU) 2020/1058 of 27 April 2020 Amending Delegated Regulation (EU) 2019/945 as Regards the Introduction of Two New Unmanned Aircraft Systems Classes: (EU) 2020/1058*; Publications Office of the European Union: Luxembourg, 2020.
46. Luftverkehrs-Ordnung—LuftVO. 2021. Available online: https://www.gesetze-im-internet.de/luftvo_2015/ (accessed on 18 November 2023).
47. Vieira da Silva, L.M.; Sieber, C.; Fay, A. Vollautomatisierte Funktionsermittlung und Schlussfolgerungen durch formale Drohnenklassifizierung. In *Automation 2023*; VDI Verlag: Düsseldorf, Germany, 2023; pp. 705–718, ISBN 9783181024195.
48. Dipul. Digital Platform for Unmanned Aviation. Available online: https://www.dipul.de/homepage/en/ (accessed on 20 November 2023).
49. Scioscia, F.; Bilenchi, I.; Ruta, M.; Gramegna, F.; Loconte, D. A multiplatform energy-aware OWL reasoner benchmarking framework. *J. Web Semant.* **2022**, *72*, 100694. [CrossRef]
50. Iovino, M.; Scukins, E.; Styrud, J.; Ögren, P.; Smith, C. A survey of Behavior Trees in robotics and AI. *Robot. Auton. Syst.* **2022**, *154*, 104096. [CrossRef]
51. Castano, L.; Xu, H. Safe decision making for risk mitigation of UAS. In Proceedings of the 2019 International Conference on Unmanned Aircraft Systems, Atlanta, GA, USA, 11–14 June 2019; IEEE: Piscataway, NJ, USA, 2019; pp. 1326–1335. [CrossRef]
52. Shang-wen, Y.; Ming-hua, H. Estimation of air traffic longitudinal conflict probability based on the reaction time of controllers. *Saf. Sci.* **2010**, *48*, 926–930. [CrossRef]