An Ontological Approach for the Dependability Analysis of Automated Systems

Guillaume Ollier, Morayo Adedjouma, Simos Gerasimou, Chokri Mraidha

An Ontological Approach for the Dependability Analysis of Automated Systems

Guillaume Ollier[1], Morayo Adedjouma[1], Simos Gerasimou[2], Chokri Mraidha[1]
[1] *Paris-Saclay University, CEA, List, F-91120, Palaiseau, France*
[2]*University of York, Department of Computer Science, United Kingdom*
guillaume.ollier@cea.fr, morayo.adedjouma@cea.fr, simos.gerasimou@york.ac.uk, chokri.mraidha@cea.fr

***Abstract*—This paper presents the Ontology Language for the Dependability of Automated Systems (OLDAS), a modeling language based on Unified Modeling Language (UML) that aims to support dependability assessment for Automated Systems (ASs), i.e., systems intended to perform a function with minimal or no human intervention. OLDAS extends the Unified Foundational Ontology (UFO) and embeds validation rules to prevent constraint violations in ASs analysis. Specifically, the paper presents how OLDAS can support different activities during the design of ASs, from the definition of the Operational Design Domain to scenario-based analysis. OLDAS is available as a plugin of the open-source Papyrus for Robotics framework.**

***Index Terms*—Autonomous Systems, Automated Driving Systems, Artificial Intelligence, Safety Engineering, ODD, ML-based Systems**

I. Introduction

The growing use of Learning-Enabled Components (LECs) to perform complex tasks in safety-critical systems brings major challenges in several industrial domains ranging from automotive and aviation to healthcare, robotics, and manufacturing. To support automation of the target system, it includes capabilities for perception and sensing based on computer vision-related tasks like scene understanding, object recognition, and event detection. According to the J3016 standard [1], such systems are called *"automated systems"* instead of *"autonomous systems"* to emphasize that they can *"depend on communication and cooperation with outside entities for important functionality (such as data acquisition and collection)"*.

Although actual deep learning-based LECs promises outstanding performances in implementing automation features in Automated Systems (ASs), establishing the dependability of the ASs integrating such components using classical risk assurance methods is impractical [2]. Since these components follow a data-driven paradigm and learn through examples, they lack formal specifications that enable establishing their compliance with safety requirements [3]. In addition, these components are non-interpretable, i.e., it is not possible to directly access the exact behavior of the model. These two limitations make it impossible for the verification and testing processes to use a small set of predefined standardized scenarios to ensure that these components will respond appropriately in all possibly encountered situations. Furthermore, specifying the infinite usage scenarios together with their influence parameters is unreachable for ASs with a complex operating environment [4]. It is then not feasible to test these systems in every possible scenario. An evaluation approach that focuses on analyzing a subset of possible scenarios as the more relevant ones with regard to safety might lead to facing unpredicted situations after deployment. To address this challenge, recent methods focus on formally defining scenario space, i.e., the set of all possible usage scenarios for an AS [5], [6]. The goal is to define an unambiguous and standardized description of necessary concepts for AS scenarios representation together with their parameters. Ensuring that such a formal specification prevents a mismatch with reality is essential to allow efficient dependability analysis. One must also provide the formal scenario-space description in a machine and human-readable format to allow knowledge transfer between human experts and systems. Such formal scenario-space specification and the above requirements that it must satisfy can be handled through an ontology, i.e., a representation of the body of knowledge in a given field with the involved concepts, the relations, categories, and properties to structure these concepts [7].

The present paper proposes Ontological Language for the Dependability of Automated Systems (OLDAS) language. The language allows the ASs' scenario-space modeling to support their development following the constraints stemming from LEC-based systems. The capabilities and limitations of the LEC can be specified by additional constraints on the scenario-space model and refined through the development process. OLDAS is defined as an extension of Unified Foundational Ontology (UFO) [8], a Unified Modeling Language (UML)-based language [9] for ontology modeling. We adapted a subset of UFO to scenario-based dependability analysis and extended it to fit the systems and software engineering practice in compliance with the ISO/IEC/IEEE 24765 [10]. OLDAS includes the concepts related to Operating Features (OFs), i.e., the environment and system's elements on which one can estimate the parameters and state values in operation; the Automation Features (AFs), i.e., the hardware and software components that implement the system automation; and hazard-related events. OLDAS augments the existing UFO modeling constraints with additional rules to correctly address hazardous scenarios' descriptions The rest of this paper is organized as follows: Section II presents some background; Section III presents the related works; Section IV presents the approach with its concepts, metamodel, usages, and implementation details; Section V presents an application of OLDAS on the definition of an Operational Design Domain (ODD) on a

quadcopter drone use case; and Section VII outlines conclusion and future works.

II. Background

A. Scenario Description for Automated Systems

To support the safety-oriented development of ASs, it is essential to follow consistent terminology regarding their behavior analysis. Ulbrich et al. (2015) [11] define the terms *"scenario"* and *"scene"*. A *"scene describes a snapshot of the environment"*, whereas a *"scenario describes the temporal development in a sequence of scenes"*. A scenario can then be described by several scenes with events (including action events) in between. The paper from Ulbrich et al. also adopt the logical and concrete scenarios categorization from Menzel et al. [12]. The *logical scenarios* represent a formal notation of the entities and their interrelationships within a scenario space. The *concrete scenarios* complete the logical representation by instantiating concrete values for each element in the scenario space.

B. Ontology Modeling Language

In the domain of knowledge representation, ontological models provide formal methods for structuring knowledge and enabling automated reasoning over it, e.g., automatic inferences and proposition satisfiability [13]. To do so, the model represents the concepts relevant to the reasoning task with their categories, properties, and relations. This formalized representation ensures human and machine readability for inference engines and interoperability with various applications. The modeled knowledge concepts can be categorized as *endurant* or *perdurant* [14]. An *endurant* concept is persistent through time and is necessarily wholly present in each time interval at which it exists, whereas a *perdurant* concept is situated in a period and only exists during this delimited period; hence, events and situations are *perdurant* concepts. One also distinguishes two categories of entities in ontology models: *individuals*, also called "objects" and *types*, also called "classes" [8], [15]. The *types* are abstract concepts, e.g., "Person", "Night", "Thing", whereas *individuals* can be concrete objects classified by the types (e.g., a specific person), or other abstract concepts (e.g., a specific number, a specific organization name). An ontological model does not specify individuals but furnishes means to classify them. Using ontological models to support systems development with automated reasoning has been widely studied [16]. The most common approach for ontology formalization is the Web Ontology Language (OWL) [17]. It enables the creation of class hierarchies and properties and the definition of instances with operations. Although the OWL language is employed in practice for conceptual modeling, the language alone cannot ensure that a scenario and reasoning that can be generated within an ontological model are possible for the target system. Upper ontologies aim to solve this limitation by adding a semantic layer to structure the knowledge model and ensure domain interoperability. Among them, the Unified Foundational Ontology (UFO)[8] proposes extending UML to take benefits of this standard modeling language. UML is a general-purpose modeling language intended to support software or system development. UML proposes several diagrams. Among them, the class diagram helps describe a system's structure with classes, attributes, operations, and relationships, which present similar semantics to ontological concepts [18]. Using a UML class diagram for UFO ontology modeling presents the advantage of relying on a wider Model-Driven Engineering community than to OWL community. However, Barcelos et al. [19] define a set of rules to automate the transformation from UFO models to OWL to ensure interoperability with OWL-based automated reasoning tools. Our approach relies on a UML profile diagram for UFO-based ontology modeling.

C. Dependability Definition

The present paper relies on the definition of the *dependability* concept from the taxonomy of Avizienis et al. [20]. The taxonomy defines *dependability* as a set of the following attributes: - availability, i.e., the readiness of correct services; 1) reliability, i.e., the continuity of correct service; 2) safety, i.e., the absence of catastrophic consequences on the user and the environment; 3) integrity, i.e., absence of improper system alterations; 4) maintainability, i.e., ability to undergo modifications and repairs. Preserving a minimum safety level for a system can require aborting certain services leaving the system in a degraded mode. For ASs, it comes down to aborting partially or entirely the automation service and initiating a safety mitigation process, e.g., user fallback request. The service degradation may be considered as a partial failure and then reduced reliability for safety preservation. Our approach then considers that safety must be dealt with the other dependability items to be monitored properly.

III. Related Work

The present section discusses prior work on the ontological approaches for the dependability analysis of safety-critical systems and the ODD formalization presented in this paper to illustrate an ASs development stage supported by OLDAS.

A. Ontological Approaches for Dependability Analysis

Bakirtzis et al. [21] propose an ontological UML metamodel for cyber-physical systems design to support their safety, security, and resilience to service disruption. Zaki et al. [22] propose an ontological approach with a logic-based model to support the runtime ability to verify that ASs are safe and reliable for operation within a dynamic environment. The OpenXOntology standard [23] addresses the description of road vehicles' environment by proposing an ontology model based on High-Quality Data Models[24], i.e., a core ontology that aims to support large-scale data integration. Several works defined UFO-based ontologies to support the safety-oriented development of cyber-physical systems. The "ESHA ontology"[25] raises a formal ontological framework adapted from UFO for hazard identification of ASs. Although this work describes a method to incorporate concepts from other metamodels for its construction, the ontology language is not

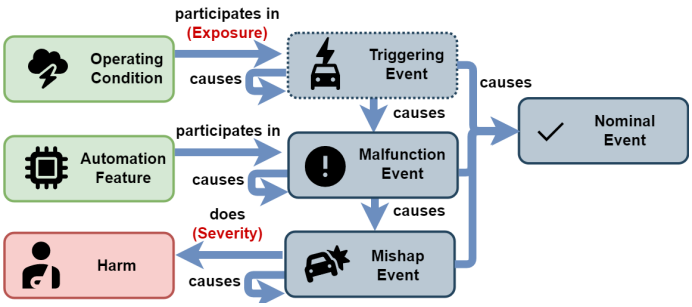


Fig. 1: Critical scenarios description.

available. Zhou et al. [26] propose Hazard Ontology (HO) that explicitly represents the hazard concepts and their relations with the system under analysis and its environment. Sales et al. [27] defines Common Ontology of Value and Risk to harmonize different existing risk concepts in the literature. Debbech et al. [28] proposes the Dysfunctional Analysis Ontology. The ontology aims to provide a terminological clarification to support dysfunctional analysis to prevent hazards during early design phases.

B. Formalization of Operational Design Domain

Schwalb et al. [29] presents a syntax and formal semantics for defining an ODD. The syntax allows importing OWL ontologies and specifying permissive and restrictive constraints on Operating Conditions (OCs). The obtained specification is simultaneously machine and human-readable. Erz et al. [30] proposes an ontology-based approach offering systematic guidance for an ODD definition using a UML model organized over five hierarchical layers of ODD attributes. The ontology encompasses automated vehicle architecture elements and provides a basis for knowledge-based scenario creation. The ASAM initiative develops OpenODD, a human-readable and a machine-readable format for ODD specification grounded on a Domain-Specific Language [31]. Thanks to well-defined syntax and semantics, the language enables using hierarchical elements to define a world-driving model. It embeds the ability to tie real-world values to these ODD attributes and means to define permissive or restrictive constraints on them.

IV. Ontological Language for the Dependability Analysis of Automated Systems

A. Concepts

The present section introduces and defines the semantics used in the meta-model.

1) Scenes and Scenarios: OLDAS needs to capture the different elements that may affect the correct AS operation into an ontology model. To achieve this, one reuses the distinction of *endurants* and *perdurants* presented in II-B. The participants of an event are the necessary endurants to enable the latter event. The ensemble of these participants defines a scene at the logical level. The scenario space of an AS can be represented at the logical level, with events trees representing all possible future scenarios for the initial scenes. A given scenario in this

Fig. 2: Agents participation in critical scenarios.

space is an events sequence from these trees with participants specified for each event. The probability of an event occurrence is correlated to the probability of the presence of its participants. The uncertainty in this probability estimation is determined by the uncertainty of the participants' detection and the uncertainty on the completeness of the events participants identified in the model.

2) Critical Scenarios: To make OLDAS suited for dependability analysis, it is important to support the description of critical scenarios, i.e., scenarios including elements that can affect the system's dependability. Fig. 1 presents the concepts used by OLDAS to describe a critical scenario at the system level. The OCs and AFs are the endurant concepts that participate in critical scenarios, and they are represented by green boxes. The harm is the resulting effect of the critical scenario and it is represented by the red box. The severity level depends on the estimated impact of this harm based on various criteria. For safety, the impact is estimated based on the possibility of human lives being endangered. For availability, the impact is measured from the outage duration of the automation service. The critical scenarios are described through a chain of events that lead to the system's harm. These events, represented in the figure in gray boxes, can be categorized into three types. The *triggering event* describes the specific conditions of a driving scenario that possibly initiate a subsequent system reaction, possibly leading to a mishap. The triggering event can be unknown, so it is represented in the figure with a dotted box. The *malfunction events* describe the failure or fault of a hardware or software component preventing it from providing its intended service. A *mishap event* is an event resulting directly or indirectly in harm. The events representing a normal condition are called *nominal events*. We call *criticality exposure level* the probability value of being in the OCs that can cause a critical event if coincident with the failure mode under analysis. This exposure value is estimated from the probability of occurrence of the OCs participating in the critical scenario's triggering event. We call *service failure severity level* the ranking value of the failure consequence upon the system and the environment. This severity value is estimated from the harm grade of a mishap event.

3) Agents and Actions: An *agent* is defined in this paper as a (human or software) element capable of acting on the environment. This actuation process is motivated by objectives and led by a reasoning process. These agents can then produce actions, and some of them are qualified *unintended* when they affect the system in a way not intended by the manufacturer of the latter system. These *unintended actions* are a particular type of *triggering event*. Fig. 2 represents the relevant agent-related concepts in yellow boxes. The participation of these unintended actions in critical scenarios is optional, so *Action Event* is represented with a dotted box on the figure.

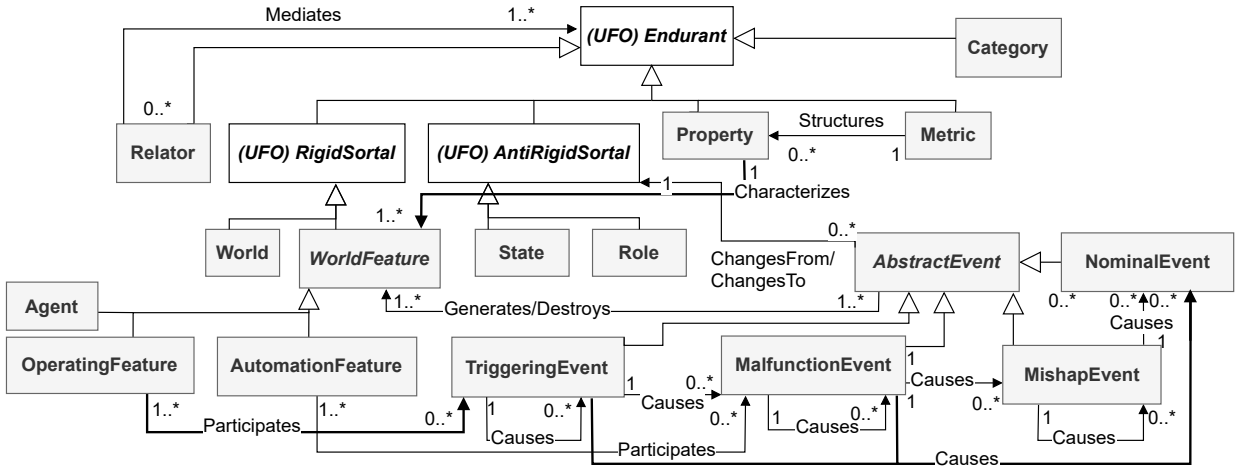


Fig. 3: The OLDAS Metamodel

4) Properties: The scenario analysis of ASs must consider the parameters perceivable from its OCs and AFs. These properties can be associated with structured value and corresponding metrics, e.g., wind speed and satellite received signal strength. Some other properties have no structured values (wet road, artificial illumination) and are only characterized by their presence or absence in a scenario.

5) States and Roles: The concepts of *states* and *roles* are the concepts instantiated by changes on, respectively, in intrinsic properties or in relational contexts. For example, the "day" and "night" conditions are states instantiated by the illuminance intensity, and the "driver" and "supervisor" are roles instantiated by the relation of a human operator with an AS. These concepts shall inherit identity providers, i.e., types that individuals retain throughout their lives and that allow their identification.

B. Metamodel

Fig. 3 presents the OLDAS metamodel. The stereotyped classes are represented in boxes, and the stereotyped associations are represented with labeled arrows. From the UFO meta-language, the *Subkind* stereotype represents the rigid specialization (i.e., the sub-classification of a concept that cannot change through time) of an identity provider. OLDAS simplifies this sub-classification by using abstraction. The stereotypes *OperatingFeature*, *AutomationFeature*, *Agent*, *Property*, *Relator*, and *Metric* are identity providers (defined in Sec. IV-A5). The *OperatingFeature* and *AutomationFeature* stereotypes represent elements on which one can measure the OC, and the system capabilities, respectively. An OLDAS model shall have one class stereotyped *World*, and all *WorldFeature* classes must have a direct or indirect composition relation with the *World* concept. The stereotype The properties with intrinsic values must be structured by a *Metric*.

The abstract stereotype *AntiRigidSortal* represents concepts instantiated by a change. It supertypes the two stereotypes *State* and *Role*. Classes stereotyped as role and state must have exactly one identity provider as an ancestor. The stereotype *Relator* represents a mediation (i.e., the intervention of a concept to describe a relationship) between at least two individuals, e.g., human-machine communication mediates at least one driver or passenger and one dashboard. Classes stereotyped *Role* must be connected to a class stereotyped *Relator*. The stereotype *Category* can be used to refactor multiple relationships among classes. Critical scenarios can be represented thanks to the stereotypes *NominalEvent*, *TriggeringEvent*, *MalfunctionEvent*, and *MishapEvent*. The stereotype *Agent* is a specialization of *WorldFeature* which can create *ActionEvent*.

C. Usages for Automated Systems' Development

OLDAS can support the AS development at different stages. The possible usages of OLDAS are presented below.

1) Specification of AS's Operational domain and Operational Design Domain: The ODD concept is introduced in the J3016 taxonomy [1] for the automotive domain and is defined as a "*Operating conditions under which a given [driving] automation or feature thereof is specifically designed to function*" [1] The definition can be generalized to any usage domain as a representation of the operating environment of the AS related to its capabilities and its current states. The ODD of an AS is described inside its Operational Domain (OD), i.e., the set of all the possible OCs for a usage domain defining its scenario space. OLDAS enables the description of OCs, their qualifiers, i.e., the elements specifying the scenario parameters. These elements are essential to describe an OD and ODD boundaries inside it.

2) Logical Scenarios Specification: For the scenario-based system analysis purpose at the early development stages, OLDAS can handle the specification of usage scenarios at the logical level, i.e., a formal description of the scenario including the OCs represented by state variables [12]. The scenario specification supported by OLDAS can help experts provide more complete scenario-space coverage for analysis and testing. On the other hand, the automated scenario interpretation

[1]Note that we remove the word "*driving*" from the J3016 ODD definition to make it applicable to all ASs.

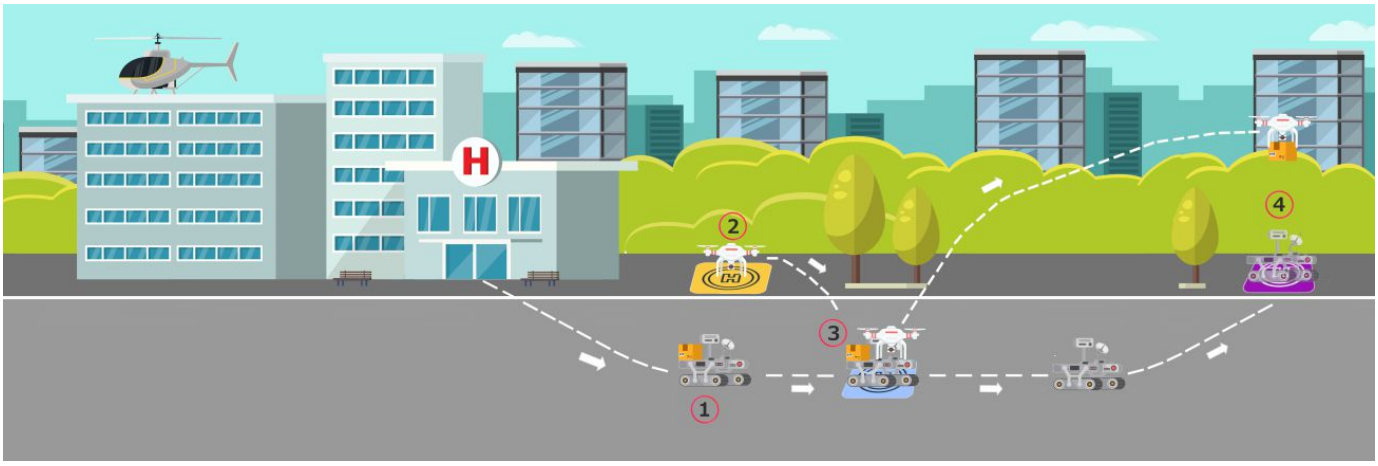


Fig. 4: Illustration of the "Logistics in Urban Areas" Use Case.

can be used to check dependability requirements depending on the detected OCs for a specific AS usage scenario.

3) Dataset Preparation: OLDAS can help standardize the data selection and labeling. It enables identifying from datasets the relevant entities, with their parameters and event occurrences. With the scenario specification ability of OLDAS, one can measure the dataset coverage regarding the OD of the AS.

4) Hazard Analysis Support: OLDAS model could support The identification of LEC hazards, their causes and effects. The profile enables specification of AFs AS capabilities. One can identify malfunction events that can affect the AS's dependability. OLDAS can also be used to represent the different AFs failures. Through the specification of AS capabilities, their failures, and scenarios, one can identify the causal chain events triggering AFs malfunctions and the resulting mishap events.

5) Runtime-Monitoring Component Design: The runtime monitoring of ASs is a promising approach to ensure the ability of such systems to deliver the intended service with an acceptable level of risk[32]. The systems' risk can be estimated using Probabilistic Graphical Models (PGMs) [33]. An OLDAS-based model can be used to define the architecture of the PGM thanks to the causal chain representation of the influence factors contributing to mishap events.

D. Implementation

OLDAS was developed as a UML profile in Papyrus [34] version 6.2.0 on Eclipse IDE version 2022-06 (4.24.0). The profile is implemented in Java and augmented with validation rules in Object Constraint Language (OCL)[35]. OCL is a declarative language compatible with the first-order logic paradigm that allows to specify constraints rules on UML models. The OLDAS profile and associated documentation are available as a plugin in the open-source Papyrus for Robotics framework [2].

V. Illustrative Case Study

A. Case Description

Figure 4 illustrates the selected use case for "Logistics in urban areas" [3]. It includes a quadcopter and a ground rover system collaborating to transport a parcel between two buildings. Typically, the rover will transport the parcel from inside the first building to the planned landing zone outside the building. Then the parcel will be transferred to the drone landed on the rover. The latter will fly to the second landing zone and do the reverse operation on a second rover. The second rover will then deliver the parcel inside the second building.

B. Quadcopter Domain Ontology Specification

An OLDAS-based quadcopter ontology is modeled using documentation compiled by experts from the different disciplines involved in the development of this type of AS. The necessary concepts for the operating environment description of quadcopters are identified from the following documents: 1) the "Guide to Meteorological Instruments and Methods of Observation" from the World Meteorological Organisation [36] for environmental conditions identification; 2) the "Supporting Safe and Secure Drone Operations in Europe" report from the SESAR Joint Undertaking [37] for geospatial information identification; 3) the UAVid [38] for urban obstacles identification; 4) the document "Canadian Civil Aircraft Register: Number of Aircraft by Category Result" [39] for low-flying aircraft agents identification. The Categories, UC parameters, metrics, states, and roles associated with the concepts remain to be identified. E.g., one may specify the state "LightRain" of the operating feature "Rainfall" with a property "RainfallIntensity" defined in the range [0, 2.5] mm/h. In order to support the hazard analysis in the ODD scenario space, one shall identify the quadcopter-related faults and their mishap consequences.

[2]OLDAS profile and model examples are available here:https://git.eclipse.org/c/papyrus/org.eclipse.papyrus-robotics.git/tree/plugins/oldas

[3]the use case was proposed in the COMP4DRONES project is presented here:https://www.comp4drones.eu/deliverables/

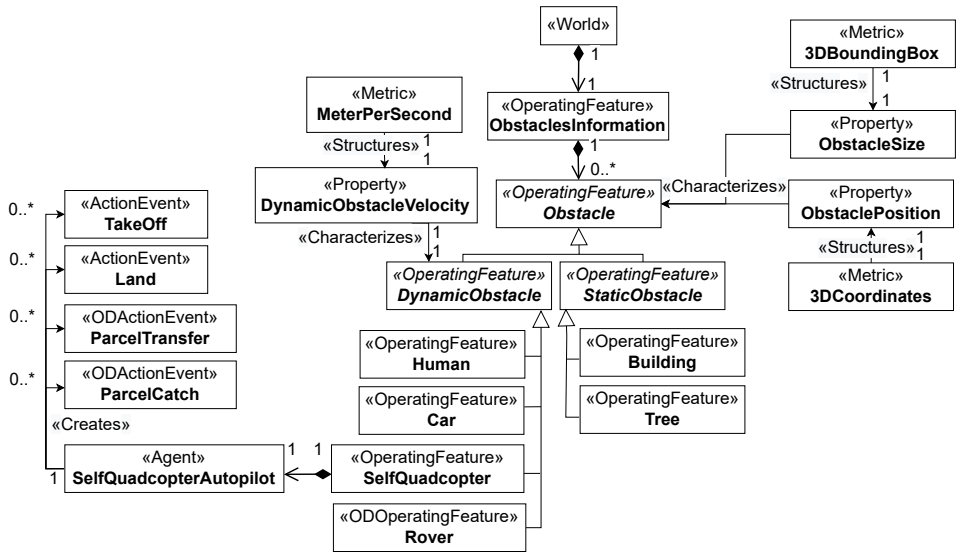


Fig. 5: Excerpt of the "Logistics in Urban Areas" Ontology Model

The CV-HAZOP checklist [40] is a fault catalog related to computer-vision components. This document populates the model with the triggering and malfunction events associated with sensing AFs (Camera and LiDAR) functions.

C. Use Case Ontology Specification

Fig. 5 presents an extended ontology structuring the OD of the use case from a subset of the quadcopter ontology. The model includes the quadcopter's potential obstacles and specific actions stereotyped with *ODActionEvent* and specific operating features stereotyped with *ODOperatingFeature*.

D. ODD Specification

The specification of ODD exit conditions is modeled using OCL. Constraint specification can concern a unique OC or a combination of OCs. Each *Metric* class contains a datatype as an attribute to specify their value with their type. For example, the following ODD constraint:

"The drone is inside its ODD when the precipitation intensity is lower than 50 mm/h (violent precipitation) or if the luminance is higher than 2000 lux (daytime)" is defined in OCL as follows:

```
context PrecipitationIntensity inv:
    self.mmh.dataType.value >= 50
    implies LuminousPower.getInstance()
            .lux.dataType.value > 2000
```

E. Logical Scenario Representation

Critical scenarios at the logical level can be represented from an OLDAS model. Fig. 6 presents an OLDAS-based basic drone ontology. The modeled domain ontology specifies environmental conditions, e.g., *"Precipitation"*, with its states *"Rainfall"* and *"Snowfall"*, and intensity parameters, agents with their possible actions, an AF (here *"ObstaclePerceptionAlgorithm"*), and a sequence of events that can lead to a mishap. In Fig. 7, we draw from this ontology two scenarios represented as instance specification models.

VI. Evaluation

In this section, we discuss an evaluation of the OLDAS profile to demonstrate that it satisfies the requirements for an ontological approach for the dependability analysis of AS. We aimed to answer the following Research Questions (RQs):

RQ1. How to enable scenario space formalization?

OLDAS allows capturing the different elements that may affect the correct AS operation into an ontology model. It uses the *EndurantType* stereotype to model these elements with the *Participates* association to model the participant endurants in events. The *Causes* associations model the cause-consequence relation between events. OLDAS-based can also specify the occurrence range of endurant individuals, i.e., indicating how many individuals of this type one can observe in a scenario for each endurant type. For example, pedestrians can be present as many times as necessary, whereas rainfall is present zero or one time. The number of occurrences of endurants with their respective properties impacts the risk and shall be considered in the dependability analysis. In addition, OLDAS supports adding new attributes and values to extend an existing ontology. Thanks to Papyrus features, it also supports combining multiple models into a new wider model. This makes it possible to integrate cross-domain knowledge (e.g., weather analysis, human operators' behavior) into the model.

RQ2. How to enable critical scenarios description?

As explained in Section IV-B, the stereotypes *TriggeringEvent*, *MalfunctionEvent*, and *MishapEvent* allow OLDAS to represent the critical scenario events. Modeling constraints ensure the causal order *trigger*, *malfunction*, and *mishap*. The *TriggeringEvent* classes must have an incoming *Participates* association from the *OperatingFeature* classes, and the *MalfunctionEvent* classes must have an incoming

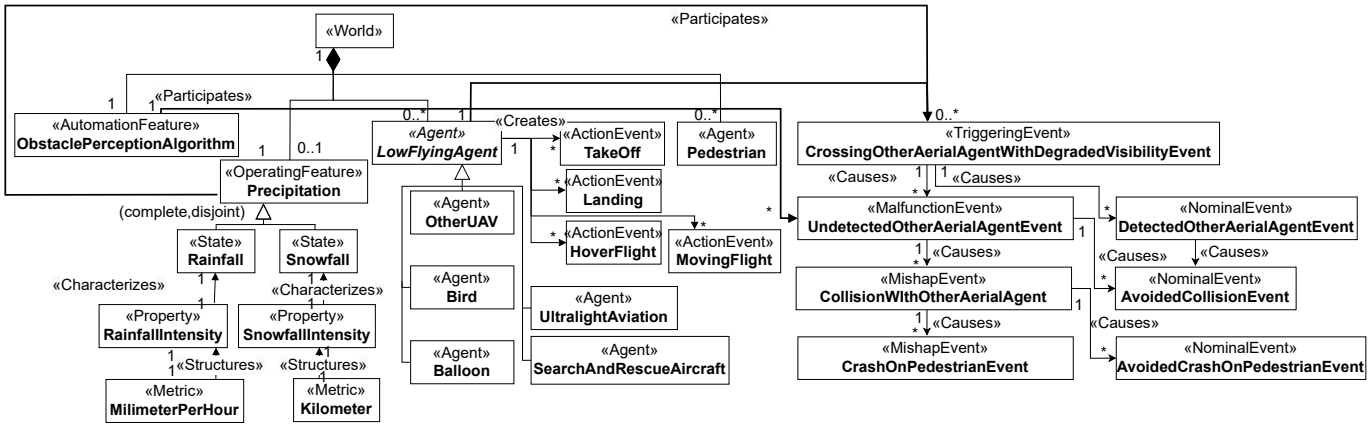
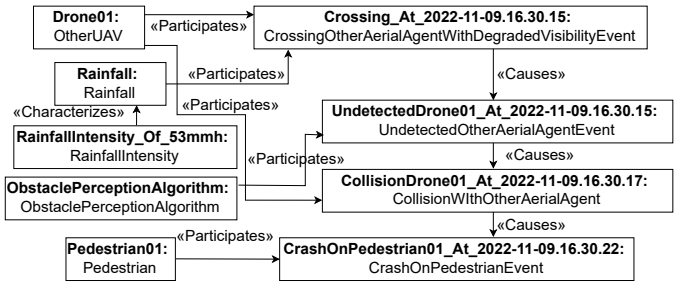
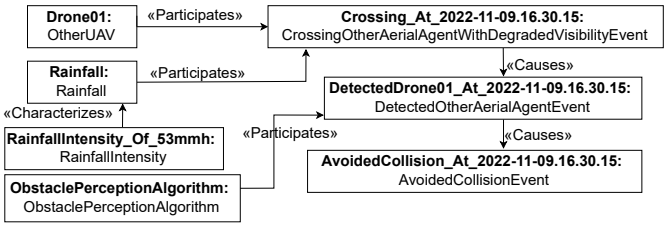


Fig. 6: OLDAS Class Diagram for Aerial Drones



(a) An aerial drone suffers perception degradation caused by violent rainfall. This leads to the false negative detection of another moving aerial agent and a collision between the drone and the other agent. This finally causes a crash on a pedestrian.



(b) An aerial drone suffers perception degradation caused by violent rainfall. The aerial drone detects another drone passing on the trajectory despite the bad visibility, and the collision is avoided.

Fig. 7: Two Instance Specifications of the OLDAS Model to Represent Logical Scenarios

Participates association from the *AutomationFeature* classes. This latter constraint ensures the specification of the OC initiating the triggering event and the AF failing their service during a malfunction. Instance specifications of OLDAS models can represent logical scenarios for the modeled AS's scenario space, as shown in Sec. V-E.

RQ3. How to provide human and machine readability?

OLDAS needs to be intended to be used by human experts. This usage needs to be easy to learn for system and software engineers, and the produced models need to be easy to read. The use of UML and the compliance with the systems and software engineering vocabulary (collected and standardized in ISO/IEC/IEEE 24765) help to satisfy this criterion. The ease of use for the human modeler also involves minimizing the number of concepts needed to grasp OLDAS. In addition, it is required to reference each modeled concept and its definitions in the taxonomy document source to prevent ambiguity and confusion on these concepts' definitions. OLDAS models also need to be parsable by coherence verification processes and should enable queries to retrieve and manipulate data stored in the model. Together, human and machine readability allows knowledge transfer between the human expert and the reasoning engine. It enables the OD and AF specification and analysis.

RQ4. How to support automated reasoning?

Numerous reasoning algorithms are developed with OWL-based inference engines, e.g., forward chaining [41] and Bayesian inference [42] algorithms for inductive logical reasoning. To make OLDAS-based models compatible with OWL-based tools for automated reasoning, we can rely on an existing UML to OWL transformation tool within Papyrus framework [43]. The transformation engine defines a mapping of UML elements and their counterparts in OWL while it addresses the coherence and preservation of the semantics and constraints coming from the UML's models. In addition, OLDAS embeds OCL-based constraint rules to ensure the correctness of ASs scenario modeling.

VII. Conclusion

This paper presents OLDAS, a UML-based modeling language to support dependability assessment for ASs. We present the OLDAS with its concepts and usages for different purposes, including logical scenario representation and ODD specification. We exemplify those usages in a drone use case. In future works, we intend to develop a tool for consistency checking on ODD constraints (i.e., checking equivalence, redundancy, and conflicts between the specified constraints).

In addition, we intend to develop an OLDAS-based Hazard Analysis and Risk Assessment and probabilistic reasoning models for uncertainty-aware runtime dependability analysis of ASs.

Acknowledgment

This work was partially supported by the PRISSMA project "Certification of AI-based autonomous mobility systems" of the Grand Défi launched by the French Innovation Council.

Acronyms

AF Automation Feature
AS Automated System
LEC Learning-Enabled Component
OC Operating Condition
ODD Operational Design Domain
OD Operational Domain
OF Operating Feature
OLDAS Ontological Language for the Dependability of Automated Systems
OWL Web Ontology Language
UFO Unified Foundational Ontology
UML Unified Modeling Language

References

[1] S. Mobilus, "SAE J3016 Taxonomy and Definitions for Terms Related to Driving Automation Systems for On-Road Motor Vehicles," Society of Automotive Engineers International, Tech. Rep., 2018.

[2] R. Salay and K. Czarnecki, "Using machine learning safely in automotive software: An assessment and adaption of software process requirements in ISO 26262," *CoRR*, vol. abs/1808.01614, 2018. [Online]. Available: http://arxiv.org/abs/1808.01614

[3] R. Salay, K. Czarnecki, M. S. Elli, I. J. Alvarez, S. Sedwards, and J. Weast, "Purss: Towards perceptual uncertainty aware responsibility sensitive safety with ml." in *SafeAI@ AAAI*, 2020, pp. 91–95.

[4] C. Amersbach and H. Winner, "Functional decomposition — a contribution to overcome the parameter space explosion during validation of highly automated driving," *Traffic injury prevention*, vol. 20, no. sup1, pp. S52–S57, 2019.

[5] International Organization for Standardization, "ISO/PAS 21448:2019 Road vehicles —- Safety of the intended functionality," Geneva, CH, 2019.

[6] J. M. Cluzeau, X. Henriquel, G. Rebender, G. Soudain, L. van Dijk, A. Gronskiy, D. Haber, C. Perret-Gentil, and R. Polak, "Concepts of design assurance for neural networks (codann)," *Public Report Extract Version*, vol. 1, pp. 1–104, 2020.

[7] T. Gruber, "Ontologies," *Encyclopedia of Database Systems*, pp. 1959–1959, 2008.

[8] G. Guizzardi, A. Botti Benevides, C. M. Fonseca, D. Porello, J. P. A. Almeida, and T. Prince Sales, "Ufo: Unified foundational ontology," *Applied ontology*, no. Preprint, pp. 1–44, 2022.

[9] O. A. Specification, "Omg unified modeling language (omg uml), superstructure, v2. 1.2," *Object Management Group*, vol. 70, 2007.

[10] International Organization for Standardization, "ISO/IEC/IEEE 24765:2017 Systems and software engineering — Vocabulary," 2010.

[11] S. Ulbrich, T. Menzel, A. Reschka, F. Schuldt, and M. Maurer, "Defining and substantiating the terms scene, situation, and scenario for automated driving," in *2015 IEEE 18th international conference on intelligent transportation systems*. IEEE, 2015, pp. 982–988.

[12] T. Menzel, G. Bagschik, and M. Maurer, "Scenarios for development, test and validation of automated vehicles," in *2018 IEEE Intelligent Vehicles Symposium (IV)*. IEEE, 2018, pp. 1821–1827.

[13] T. Berners-Lee. Semantic web on xml, slide 10. [Online]. Available: https://www.w3.org/2000/Talks/1206-xml2k-tbl/slide10-0.html

[14] I. Johansson, "Qualities, quantities, and the endurant-perdurant distinction in top-level ontologies." in *Wissensmanagement*, 2005, pp. 543–550.

[15] G. Booch, *Object oriented design with applications*. Benjamin-Cummings Publishing Co., Inc., 1990.

[16] C. Partridge, A. Mitchell, A. Cook, J. Sullivan, and M. West, "A survey of top-level ontologies-to inform the ontological choices for a foundation data model," 2020.

[17] H. Knublauch, D. Oberle, P. Tetlow, E. Wallace, J. Pan, and M. Uschold, "A semantic web primer for object-oriented software developers," *W3c working group note, W3C*, 2006.

[18] S. Cranefield and M. Purvis, "Uml as an ontology modelling language," 1999.

[19] P. P. F. Barcelos, V. A. dos Santos, F. B. Silva, M. E. Monteiro, and A. S. Garcia, "An automated transformation from ontouml to owl and swrl." *Ontobras*, vol. 1041, pp. 130–141, 2013.

[20] A. Avizienis, J.-C. Laprie, B. Randell, and C. Landwehr, "Basic concepts and taxonomy of dependable and secure computing," *IEEE Transactions on Dependable and Secure Computing*, vol. 1, no. 1, pp. 11–33, 2004.

[21] G. Bakirtzis, T. Sherburne, S. C. Adams, B. M. Horowitz, P. A. Beling, and C. H. Fleming, "An ontological metamodel for cyber-physical system safety, security, and resilience coengineering," *CoRR*, vol. abs/2006.05304, 2020. [Online]. Available: https://arxiv.org/abs/2006.05304

[22] O. Zaki, M. Dunnigan, V. Robu, and D. Flynn, "Reliability and safety of autonomous systems based on semantic modelling for self-certification," *Robotics*, vol. 10, no. 1, p. 10, 2021.

[23] ASAM OpenXOntology Concept Paper. [Online]. Available: https://www.asam.net/standards/asam-openxontology/

[24] M. West, "Part 4 the hqdm framework schema," in *Developing High Quality Data Models*, M. West, Ed. Boston: Morgan Kaufmann, 2011, pp. 199–200. [Online]. Available: https://www.sciencedirect.com/science/article/pii/B978012375106500021X

[25] C. Harper and P. Caleb-Solly, "Towards an ontological framework for environmental survey hazard analysis of autonomous systems." in *SafeAI@ AAAI*, 2021.

[26] J. Zhou, K. Hänninen, K. Lundqvist, and L. Provenzano, "An ontological interpretation of the hazard concept for safety-critical systems," in *The 27th European Safety and Reliability Conference ESREL'17, 18-22 Jun 2017, Portoroz, Slovenia*, 2017, pp. 183–185.

[27] T. Prince Sales, F. Baião, G. Guizzardi, J. Almeida, N. Guarino, and J. Mylopoulos, "The common ontology of value and risk," 09 2018.

[28] S. Debbech, S. C. Dutilleul, and P. Bon, "An ontological approach to support dysfunctional analysis for railway systems design." *J. Univers. Comput. Sci.*, vol. 26, no. 5, pp. 549–582, 2020.

[29] E. Schwalb, P. Irvine, X. Zhang, S. Khastgir, and P. Jennings, "A two-level abstraction odd definition language: Part ii," in *2021 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*. IEEE, 2021, pp. 1669–1676.

[30] J. Erz, B. Schütt, T. Braun, H. Guissouma, and E. Sax, "Towards an ontology that reconciles the operational design domain, scenario-based testing, and automated vehicle architectures," in *2022 IEEE international systems conference (SYSCON)*. IEEE, 2022, pp. 1–8.

[31] Asam openodd project details. [Online]. Available: https://www.asam.net/index.php?eID=dumpFile&t=f&f=4544&token=1260ce1c4f0afdbe18261f7137c689b1d9c27576

[32] P. Feth, "Dynamic behavior risk assessment for autonomous systems," 2020. [Online]. Available: https://publica-stage.fraunhofer.de/handle/publica/283031

[33] E. Asaadi, E. Denney, and G. Pai, "Quantifying assurance in learning-enabled systems," in *International Conference on Computer Safety, Reliability, and Security*. Springer, 2020, pp. 270–286.

[34] A. Lanusse, Y. Tanguy, H. Espinoza, C. Mraidha, S. Gerard, P. Tessier, R. Schnekenburger, H. Dubois, and F. Terrier, "Papyrus uml: an open source toolset for mda," in *Proc. of the Fifth European Conference on Model-Driven Architecture Foundations and Applications (ECMDA-FA 2009)*. Citeseer, 2009, pp. 1–4.

[35] Object Management Group, "Object constraint language, version 2.4," Tech. Rep., 2014. [Online]. Available: https://www.omg.org/spec/OCL/2.4/PDF

[36] WMO, "Guide to instruments and methods of observation," *World Meteorological Organization WMO*, 2018. [Online]. Available: https://library.wmo.int/doc_num.php?explnum_id=11612

[37] SESAR 3 Joint Undertaking, "U-space - supporting safe and secure drone operations in europe," Tech. Rep., 2020.

[38] Y. Lyu, G. Vosselman, G.-S. Xia, A. Yilmaz, and M. Y. Yang, "Uavid: A semantic segmentation dataset for uav imagery," *ISPRS journal of photogrammetry and remote sensing*, vol. 165, pp. 108–119, 2020.

[39] Summary of the canadian civil aircraft register - february 2018. [Online]. Available: https://wwwapps.tc.gc.ca/Saf-Sec-Sur/2/CCARCS-RIACC/smACtRes.aspx?ym=201802

[40] O. Zendel, M. Murschitz, M. Humenberger, and W. Herzner, "Cv-hazop: Introducing test data validation for computer vision," in *Proceedings of the IEEE International Conference on Computer Vision*, 2015, pp. 2066–2074.

[41] H. Rueß and S. Burton, "Safe AI–how is this possible?" *arXiv preprint arXiv:2201.10436*, 2022.

[42] Z. Ding, Y. Peng, and R. Pan, "A bayesian approach to uncertainty modelling in owl ontology," Maryland Univ Baltimore Dept Of Computer Science And Electrical Engineering, Tech. Rep., 2006.

[43] L. P. Medinacelli, F. Noyrit, and C. Mraidha, "Augmenting model-based systems engineering with knowledge," in *Proc. of the 25th International MODELS conference*, T. Kühn and V. Sousa, Eds. ACM, 2022, pp. 351–358.