A Reference Ontology for Collision Avoidance Systems and Accountability

David Martín-Lammerding
Department of Stats. Comput. Sci. Math
Public University of Navarre (UPNA)
Pamplona, Spain
email:david.martin@unavarra.es

José Javier Astrain
Department of Stats. Comput. Sci. Math
Public University of Navarre (UPNA)
Pamplona, Spain
email:josej.astrain@unavarra.es

Alberto Córdoba
Department of Stats. Comput. Sci. Math
Public University of Navarre (UPNA)
Pamplona, Spain
email:alberto.cordoba@unavarra.es

***Abstract*—Unmanned Aerial Systems (UASs) are deployed in Intelligence, Surveillance, and Reconnaissance (ISR) applications with less cost and more flexibility rather than manned aircraft. An increasing number of UAS missions requires an improvement of their safety capabilities by equipping them with Collision Avoidance Systems (CASs). It is recognized that the use of small UAS at lower altitudes is now a driving force of economic development, but a safety risk when its number increases. UAS generates heterogeneous data from multiple sources, like the Flight Control Unit (FCU), the Global Navigation Satellite System (GNSS), a radio receiver, an onboard-camera, etc. Each CAS implementation receives this data and processes it to avoid collisions. There are many CAS implementations, but each one has a specific design and data repository structure. There is a lack of standards that simplify their development and homologation. This paper presents a reference knowledge model for any CAS for UAS implemented as a novel application ontology called Dronetology-cas. It transforms data to knowledge by combining heterogeneous telemetry and onboard-sensor data using linked-data and an ontology for semantic interoperability across heterogeneous UAS traffic management systems. Dronetology-cas provides a unified semantic representation within an ontology-based triplet store designed to run in a low cost computer. Its semantic model provides advantages, such as interoperability between systems, machine-processable data and the ability to infer new knowledge. It is implemented using semantic web standards, which contribute to simplify an operational safety audit.**

***Keywords*—Semantic reasoning, ontology, UAS, knowledge, conflicts, anti-collision, sensor, embedded, air traffic.**

I. Introduction

The use of Unmanned Aerial Systems (UASs) improves efficiency in logistics applications, infrastructure inspection, emergency situations, etc. and avoids pilot risk. However, their flights are limited to certain areas of the airspace to avoid encountering other aircraft. Air traffic management must evolve to allow the introduction of large numbers of mass-market UAS. Each UAS must be equipped with new safety systems, like Collision Avoidance Systems (CASs).

CASs are developed to detect airplanes in airspace, to discover potential collision hazards and to perform maneuvers to avoid collisions. An increased use of UAS requires autonomous capabilities for safety purposes. However, UAS autonomy involves ensuring accountability. The accountability principle requires UAS operators to take responsibility for what their UAS do in a mission and how they comply with traffic management authorities. UAS operators must have appropriate records to be able to demonstrate their compliance. The accountability of an UAS flight must be ensured because any incident or accident must be able to be investigated by surveyors or authorities. In the worst case, a collision may occur, which must be investigated to determine the cause and to improve CAS.

CAS design factors are showed in Figure 1. Multiple CAS's typologies can be obtained combining different design factors. CAS should not depend on pilots or communications with centralized systems, as any delay in making a decision increases the risk of collision.

Each CAS studied has its own internal data implementation with specific structures. So, data generated by CAS have proprietary formats that are not easily inter-operable. To solve this issue we are integrating and structuring data from CAS using ontologies, linked data, and semantic integration techniques.

Ontologies [2] are formal and explicit specifications of certain domains and are shared between large groups of stakeholders. These properties make ontologies ideal for machine processing and enabling inter-operation. The use of standards for data encoding, structuring and description simplifies an audit task.

In this paper, we present a novel ontology, denoted Dronetology-cas [3], that is suitable for structuring any data generated in a CAS. Dronetology-cas includes a Knowledge Base (KB) which consists of triplets of data collected and inferred knowledge during the UAS mission.

The rest of the paper is structured as follows. Section II presents the state of the art of CAS and accountability systems, Section III defines the problem statement and Section IV describes our contribution. The ontology design is presented in Section V. Section VI formulates ontology Competency Questions (CQs) and Section VII summarizes experimental simulations results. Section VIII presents the conclusions and references end the paper.

II. Related Work

The use of UAS for data gathering is becoming increasingly widespread thanks to high quality and cost-effective sensors. Therefore, the Semantic Sensor Network (SSN) [4] ontology

 ISBN: 978-1-61208-888-4

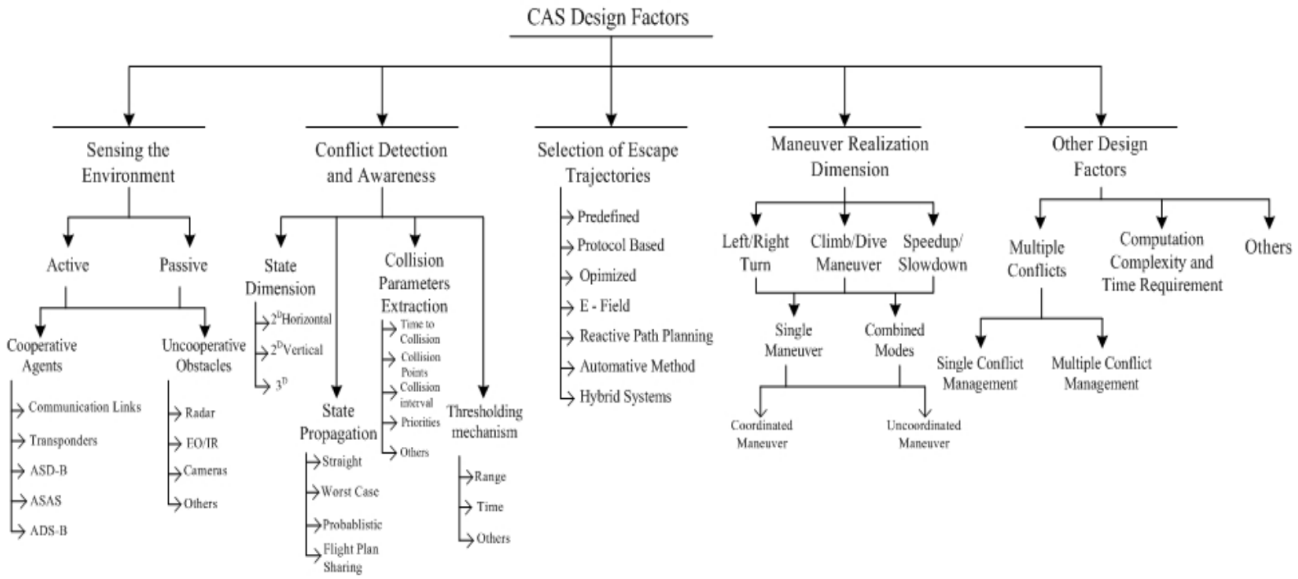

Figure 1. CAS design factors. Taken from [1].

can be used to model UAS as sensors. However, it has the limitation of not having concepts to model the UAS mission.

[5] applies semantic technologies to air traffic in order to unify heterogeneous data from multiple sources. The ontology implementation is performed centralized. However, our proposal is a decentralized ontology implemented in each UAS to serve as a knowledge base for any CAS.

[6] presents a *light-weight* ontology for embedded systems whose design reduces concepts, complexity and query times, compared to the SSN ontology. It is intended for the sensor domain and therefore has limitations for modeling an UAS.

ACAS-Xu [7] and Daidalus [8] are two reference CAS implementations whose source code is available for review. Each CAS requires a specific configuration for the same scenario. Given the same scenario, their output formats are different as shown in [9]. A limitation of both CAS is that they do not share a common conceptual model.

The accountability of an UAS flight must be ensured because any incident or accident should be able to be investigated by surveyors or authorities. There are systems similar to black boxes for UAS, [10]–[12]. They store the UAS's route and the CAS's status. However, the decision-making process prior to a maneuver is complex and its recording is not provided in these systems.

III. Problem statement

The data required by a CAS depends on how the main design factors presented in Figure 1 are combined. The main concepts of CAS used in the design of Dronetology-cas are described below.

A conflict between two UAS occurs when minimum separation, defined as the protection distance d_p, is lost. Figure 2 shows a conflict between *local UAS* and *remote UAS*. A loss of separation does not always predict a future collision, but it is a key safety indicator. A CAS deployed in an UAS is aimed at maintaining a minimum safe separation between UASs. Once a conflict is detected, a CAS diverts the UAS to a new safe path. The number of simultaneous conflicts are denoted as N_C. Time to collision t_{tc} is the time required to collide two UAS if an UAS continues at their current speed and on the same path. Lower t_{tc} values correspond to higher risk of collision. It is used to prioritize conflicts. Very Low Level airspace (VLL) is the space below 500 ft. above ground level. It is the part of the airspace intended for new UAS applications and it will concentrate most UAS conflicts.

CASs are based on different technologies that collect data from the surroundings using sensors and/or collaborative elements based on radio receivers/transmitters. UAS can deploy collaborative elements and non-collaborative sensors. A collaborative element receives and transmits position and bearing data with any other element within its coverage. Automatic Dependent Surveillance – Broadcast (ADS-B) [13] is one of the standards for collaborative systems, based on sharing location information obtained from the Global Positioning System (GPS). A non-collaborative sensor detects obstacles without any external system. There are multiple technologies applied to non-collaborative systems, such as vision cameras [14], LIDAR [15], SONAR [16], Radar [17], etc. [18] details the main technologies applied to sensors for conflict detection.

Most CASs for UASs are distributed, so they run in an onboard computer. However, the size of the UAS limits the weight of the payload, which limits the type and power of processor that can be used. Any software component used in a distributed CAS implementation should be non-compute-intensive to ensure effective real time performance.

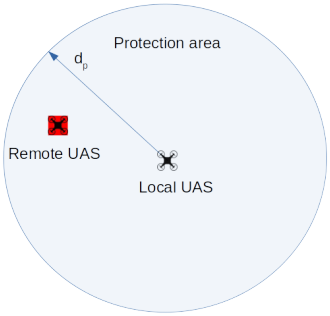

Figure 2. Conflict between local UAS and remote UAS.

IV. CONTRIBUTION

Dronetology-cas is a novel ontology intended for UAS, whose domain is anti-collision knowledge management and air safety compliance. It provides knowledge-based conflict resolution capabilities. Dronetology-cas is defined as a reference model that improves communications, inter-operation and automation of some air traffic management tasks.

Dronetology-cas defines the foundations to implement a knowledge-based CAS. It provides two modes of integration with a CAS: *repository mode* or *knowledge mode*. The *repository mode* stores data in a semantic structure, so others systems can understand and use it. The *knowledge mode* provides additional knowledge using reasoning from current data.

Dronetology-cas offers key advantages over other repository or log storage implementations. This is achieved by the web semantic technologies used in its implementation. Dronetology-cas key features are performance, modifiability, ease of maintenance, built-in inference capabilities and potential for reuse.

V. DRONETOLOGY-CAS: THE APPLICATION ONTOLOGY

Dronetology-cas is an application ontology derived from the domain ontology Dronetology [19]. The domain of Dronetology is UASs. Dronetology-cas formal specification is based on the design factors shown in Figure 1.

A. Dronetology: The domain ontology

The purpose of Dronetology is to describe concepts that define the components of any UAS, the missions it performs and the environment that surrounds it. Its main applications are the management of bill of materials, the improvement of flight efficiency and autonomous decision making. Dronetology imports external ontologies to avoid repeating concepts from other domains. Another advantage of importing widespread ontologies is that there are data repositories (sources in Resource Description Framework (RDF) format) designed with these models that can be integrated into Dronetology.

B. Dronetology-cas description

We derive the Dronetology-cas application ontology from the Dronetology domain ontology. The domain of Dronetology-cas is CAS for UAS. The aim of Dronetology-cas is to be the KB of any CAS implementation. Therefore, Dronetology-cas is generic and extensible. Dronetology-cas simplifies auditing the CAS decision making process. Its design allows queries in the KB history to retrieve the CAS status at different times. The KB stores the temporal evolution of conflicts with other UAS and the status of the CAS. Dronetology-cas consists on a KB where knowledge is stored. It also has an inference engine that generates new knowledge by applying semantic rules to the KB. The rules are expressed in SPARQL Protocol and RDF Query Language (SPARQL) statements [20], [21]. Rules inference a conflict's attribute, an evasive trajectory method, a maneuver attribute, etc. Knowledge is obtained from data recollected from sensor systems and collaborative elements. Data sources are sensors, the Flight Control Unit (FCU) and the Global Navigation Satellite System (GNSS). Inference improves the CAS decisions thanks to knowledge derived from the data.

A CAS runs in a loop with a operation frequency. This is modeled in Dronetology-cas with the concept of *Iteration*. Dronetology-cas stores CAS status, UAS telemetry and conflicts for each *Iteration* to audit the system. Data collected from sensors are also related to the *Iteration* to provide a complete picture of the environment and the CAS. Dronetology-cas simplifies the integration of data from different sources. It integrates data from any sensor system by defining generic classes, which are not directly dependent on the technology and the implementation. These classes are *NoCollaborativeData* and *CollaborativeData* and both extend *InputData*.

A CAS estimates future positions of conflicts to obtain a maneuver that avoids a collision. In *knowledge mode*, the CAS asks the KB for knowledge to perform a specific function, such as selecting the method to estimate the position. Figure 4 shows several methods to estimate the position. Dronetology-cas stores data about conflicts and also interrelates this data to discover new connections and knowledge. This knowledge can be used to improve the prediction method of the conflict's location. For example, if the conflict has been detected only through a vision camera, the uncertainty about the heading of the conflict is higher, so the most appropriate method of estimating may be the *Worst Case* method. However, if the conflict has been detected by a collaborative element, the heading is known and there is less uncertainty. In this case, the *Straight Projection* method is the most appropriate.

When the CAS makes a maneuver to avoid a collision, Dronetology-cas stores every UAS position and groups them with a individual of class *Maneuver*. Thus, Dronetology-cas replaces multiple specific-maneuvers concepts, like *left-turn*, with a set of positions, which allows any combination of trajectories, altitudes and speeds. Full trajectory prediction made by the CAS are not stored in Dronetology-cas. The dynamics of 3D conflicts are modeled in Dronetology-cas as different positions at different times.

 ISBN: 978-1-61208-888-4

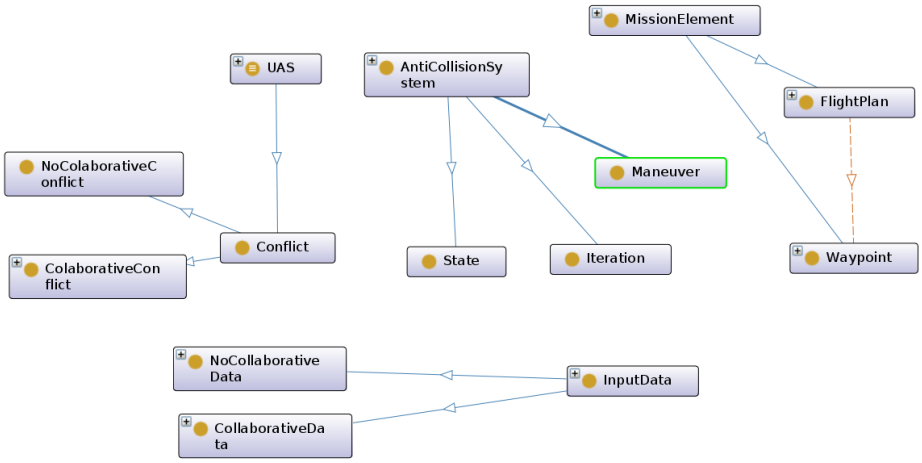

Figure 3. Dronetology-cas main classes

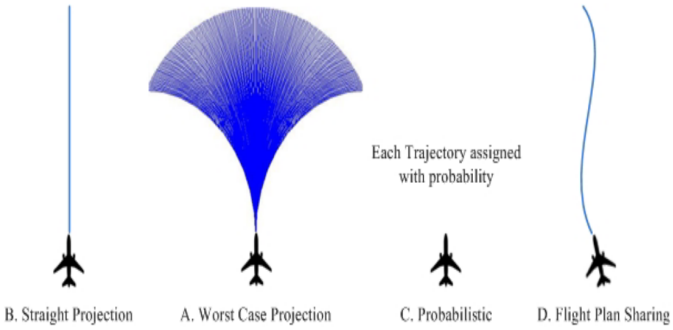

Figure 4. Methods used for projecting current encounter's information. Taken from [1].

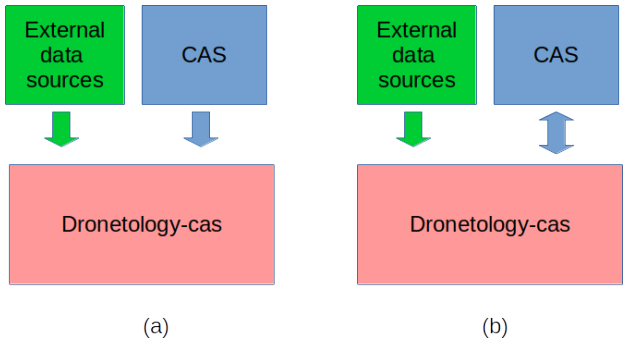

Figure 5. Dronetology-cas integration alternatives: (a) repository mode (b) knowledge mode

C. Dronetology-cas integration with a CAS

Any CAS can integrate Dronetology-cas in two ways: *repository mode* or *knowledge mode*. The Dronetology-cas repository mode requires that the CAS implements some data source specific code to translate UAS data from its original source format to Dronetology-cas ontology triplets. In this way, Dronetology-cas stores any conflict's data obtained from the onboard sensors into the KB.

The *repository mode* integration implies that the CAS inserts data as triplets into the KB. The CAS stores data in the KB, but it does not query it. Data stored are available for any audit process. External data sources, like Ground Surveillance Radars (GSRs), can add additional conflicts to the KB not detected by onboard sensors, although they depend on network connectivity during the UAS flight. The *knowledge mode* extends the features of the *repository mode*. It adds implicit knowledge inference and reasoning capabilities to some CAS functions, such as conflict detection or new path selection. In the *knowledge mode* integration, the CAS inserts data in the KB and also performs queries. These queries enhance CAS functions, such as classifying conflicts, prioritizing conflicts, selecting a trajectory calculation techniques according to the type of conflicts, etc. The CAS queries the KB for a specific result depending on its decision making implementation requirements. Figure 5 shows the relation between the CAS and Dronetology-cas for each integration mode.

Dronetology-cas is defined using the Web Ontology Language (OWL) language [22]. The main languages used to develop CAS (C, C++, Python) have implementations to process RDF triples [23] and ontologies in OWL format.

D. Dronetology-cas design

Dronetology-cas is an application ontology whose concepts are taken from CAS. The CAS design factors shown in Figure

 ISBN: 978-1-61208-888-4

1 are also considered. It is accessible at [3].

In order to integrate Dronetology-cas with a CAS, Dronetology-cas's concepts are defined with a high level of abstraction. The first design factor is the type of onboard sensors. They are classified into collaborative and non-collaborative sensors. Dronetology-cas models every onboard sensor as an abstract data source, instead of defining detailed concepts related to *sensors*.

Another design aspect to be considered of CAS is the method used to detect conflicts. The main differences between them are the data needed and the criteria followed to classify a nearby UAS as a conflict. Dronetology-cas integrated in *repository mode* stores the CAS and the conflict status. In *knowledge mode*, it improves the CAS capabilities for conflict-classification aggregating data from multiple sensors or linking the conflict detection sensor with the conflict estimated path. When a conflict's attribute are not available, like speed, it can be inferred from the conflict past locations. The inference of conflict attributes also improves the CAS decisions.

Finally, the method to calculate an evasive trajectory and the associated maneuver are a CAS's design choice. Dronetology-cas integrated in *repository mode* stores a maneuver as a sequence of UAS locations. In *knowledge mode*, the CAS could query Dronetology-cas to select a maneuver calculation method using knowledge about the conflict.

Dronetology-cas has been designed considering the computational limitations of onboard systems. Thus, memory usage has been reduced by limiting the number of classes in the model and avoiding importing auxiliary ontologies.

The main classes of Dronetology-cas are *UAS*, *MissionElement*, *InputData*, *AntiCollisionSystem* and *Conflict*. Figure 3 shows the main Dronetology-cas classes.

TABLE I
DRONETOLOGY-CAS COMPETENCY QUESTIONS

CQ_1	How many conflicts are detected?
CQ_2	Which UAS has the highest priority among the UAS in conflict?
CQ_3	Which conflict has the shortest time to collision?
CQ_4	Has the number of conflicts increased or decreased?
CQ_5	How has been detected the conflict with a given UAS?
CQ_6	How long it has taken to resolve a conflict?
CQ_7	Has the distance flown been increased with respect to the flight plan?
CQ_8	In which locations have there been conflicts?
CQ_9	Where and when was the collision?
CQ_{10}	How many UAS were in conflict before the collision?
CQ_{11}	What UAS has it collided with?
CQ_{12}	What maneuver was the UAS performing before the collision?

The class *UAS* describes unmanned aircrafts including the communication systems and the ground base. The class *Conflict* is a subclass of *UAS* so in our model only UAS can be conflicts. *MissionElement* is a class that enclose all the elements of a mission. The classes *Waypoint* and *FlightPlan* derive from *MissionElement*.

The class *InputData* represents any data collected from a sensor (non-collaborative), from a collaborative element (radio receiver), from the GNSS or from the FCU. The concepts *NoColaborativeData* and *ColaborativeData* are derived from *InputData* to identify a conflict and its source type. The property *drone:detect* is an object property that relates individuals of *NoColaborativeData* or *ColaborativeData* with individuals of *Conflict*.

Some classes in Dronetology-cas have geographic data defined as datatype properties. The latitude and longitude are relative to the World Geodetic System 1984 (WGS84) coordinate system. The altitude is relative to Mean Sea-Level (MSL). To improve interoperability, the Conflict class uses *geo:wktLiteral* datatype with a WGS 84 geodetic latitude-longitude. This allows Dronetology-cas to implement a geospatial web service that could be reused and recombined to fulfill a user query.

The class *AntiCollisionSystem* groups elements of any CAS. The classes *State*, *Maneuver*, *NextIterationLocation* and *Iteration* are derived from it. The state of the CAS are represented as instances of the class *State* with an attribute that codifies the state and a timestamp. The class *Iteration* relates all the knowledge stored in the KB at an instant of time.

The class *Maneuver* defines a set of locations of the UAS when the CAS is active. CAS calculates multiple location alternatives of the UAS to avoid the collision and stores them in the KB as instances of the class *NextIterationLocalUASLocation*. It also selects one locations that best resolve the conflict from the previous set of locations. An individual of class *Maneuver* groups every individual of class *NextIterationLocalUASLocation* through an object-property. Every iteration, the CAS sends to the FCU the individual of class *NextIterationLocalUASLocation*.

VI. COMPETENCY QUESTIONS

We define a set of CQs that specify what knowledge has to be entailed in Dronetology-cas. This questions has been used to validate Dronetology-cas. Some CQs are suitable for an UAS mission audit process. Others can assist the CAS in a decision making process, when Dronetology-cas is integrated in *knowledge mode*. Table I shows a list of some CQs considered. There are CQs that are intended to find out how the conflict has been resolved, e.g., CQ_6, CQ_7 and CQ_8. Some CQs help to find out what happened and how when a collision happens, e.g. CQ_9, CQs CQ_{10}, CQs CQ_{11} and CQ_{12}.

In a *knowledge mode* integration, the CAS uses the results of some CQs to make decisions. An example is the CQ *What type of conflict is X?*. With this knowledge about the conflict, the CAS selects the most appropriate way of calculating the future position of the conflict. Other CQs are intended for a security audit of the CAS. An example is the CQ to check when a collision occurred.

VII. PERFORMANCE EVALUATION

The performance of Dronetology-cas is analyzed executing CQs translated to SPARQL in a low cost computer, Pi3 [24]. We simulate a system CAS with a software component developed in Java 8 that inserts triplets with conflicts data in the KB. Response time and memory footprint are measured with different number of triplets stored in the KB. Memory footprint has been measured using the Java 8 API. The number

 ISBN: 978-1-61208-888-4

TABLE II
RESPONSE TIME (IN MILLISECONDS) AND MEMORY FOOTPRINT (IN KILOBYTES) OF REPOSITORY MODE AND KNOWLEDGE MODE IN A PI3.

No triplets	*Repository mode*								*Knowledge mode*							
	CQ_1				CQ_3				CQ_5				CQ_6			
	Response time		Memory footprint		Response time		Memory footprint		Response time		Memory footprint		Response time		Memory footprint	
	mean	sdev	mean	sdev	mean	sdev	mean	sdev	mean	sdev	mean	sdev	mean	sdev	mean	sdev
100	18.95	4.13	5025.28	1418.13	26.68	7.83	5101.67	1420.78	19.98	5.91	5104.45	1420.83	18.21	7.45	5100.81	1421.06
250	23.54	2.26	5200.98	1308.69	23.38	2.17	5241.25	1308.82	24.12	4.18	5211.77	1308.21	24.03	2.23	5233.40	1305.41
500	38.10	2.97	5441.26	1327.53	38.14	3.10	5441.51	1322.02	52.37	18.03	5377.53	1327.33	39.05	3.56	5491.02	1322.82
1000	74.42	17.78	5986.61	1332.65	67.47	3.10	5983.91	1336.00	68.49	3.18	6061.40	1331.63	68.74	3.10	5969.27	1319.01
2500	165.62	36.42	6997.64	1342.78	171.69	46.86	3362.77	1727.86	173.87	55.31	3875.86	1768.38	160.60	6.81	6972.95	1315.36
5000	320.98	64.36	6004.34	1718.44	320.30	63.49	5391.63	1587.92	355.39	100.29	5309.72	1510.33	324.63	65.44	5935.41	1724.54
10000	662.00	162.01	9545.00	1744.46	662.24	164.92	8440.54	2154.39	654.65	151.11	8560.70	2140.84	670.81	162.15	9595.25	1752.67

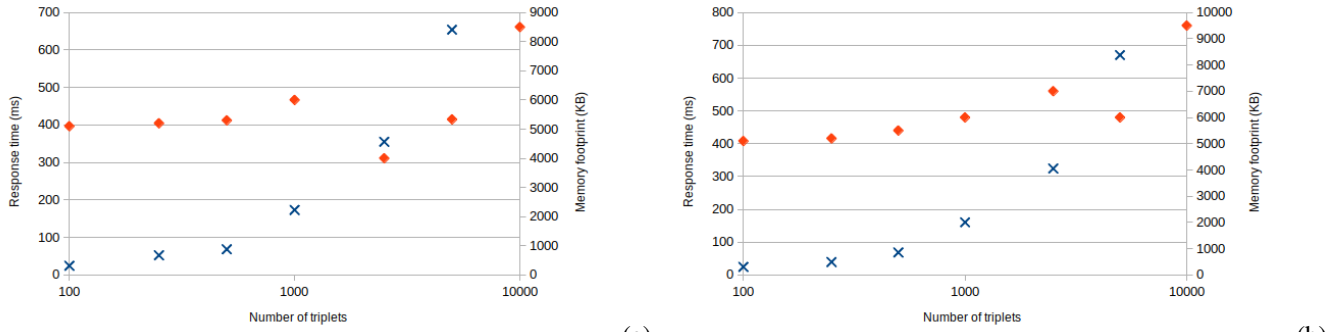

Figure 6. Response time (x) and memory footprint (♦) for *knowledge mode* for CQ_5(a) and CQ_6(b).

of triples with conflicts and CAS data in the KB grows as the UAS flies. Therefore, the flight duration determines the number of triples stored in the KB. In our tests we have simulated up to 10000 triples corresponding to 15 minutes of flight by inserting an average of 10 triples per second.

To measure response times and memory footprint, the most generic CQs have been selected as they are the most likely to be used in any integration mode. CQ_1 and CQ_3 are necessary for any auditing process to review conflicts and their status. CQ_5 and CQ_6 provide knowledge that the CAS can use to modify its response to conflicts. 100 repetitions of each case were performed to calculate the mean and standard deviation. The results obtained from the response times and memory footprint are shown in Table II. CQs considered are translated to SPARQL, available at [25].

The response time affects the CAS depending on the integration type chosen. In *repository mode*, there are no strict response time requirements as it is not required a real-time operation. However, in *knowledge mode*, the response time delays the CAS decisions. For our purpose, a suitable response time should allow to take a decision with the most recent data, before new data is available, that is, the response time should be below the refreshing rate of incoming data. Each sensor system has its refreshing rate ranging from 1 Hz of ADS-B until 20 Hz of a vision camera [26]. The response times of CQ_5 and CQ_6 obtained comply with the previous criteria as long as the number of triplets are below approximately 1000 triplets.

Figure 6 shows that Dronetology-cas response time increases when the number of triples increases. Memory consumption grows as the UAS flies as well. That is, the duration of the UAS flight increases the response time. The worst response time is at the end of a flight. This result is due to our limited implementation of the software components that instantiates and queries the KB. An option to scale up is to have two instances of Dronetology-cas model, each with a different purpose, one instance for the *repository mode* and the other for the *knowledge mode*. The instance for the *repository mode* should store all triplets, but the instance for the *knowledge mode* should keep only triplets needed for the inference process.

VIII. CONCLUSION

In this paper, we described the Dronetology-cas ontology as a value-added component for any CAS. Dronetology-cas integration modes facilitate its application in any CAS. A production-ready implementation of Dronetology-cas should take into a account the performance results and the integration mode required to balance response time and memory consumption.

As the need for UAS safety compliance is expected to increase, reference CAS implementations promoted by government agencies, like Daidalus [8], are candidates to implement advanced audit systems like the proposed in this paper.

Future work will be focused on the implementation of a CAS for UAS using Dronetology-cas and the integration of Dronetology-cas with an existing CAS. Another line of work is to create a *dataset* with semantic mission data to be used for research of UAS air traffic. Further developments of this work have the potential to achieve an ontology standard for autonomous UAS.

 ISBN: 978-1-61208-888-4

REFERENCES

[1] B. Albaker and N. Rahim, "A survey of collision avoidance approaches for unmanned aerial vehicles," in *2009 international conference for technical postgraduates (TECHPOS)*, IEEE, 2009, pp. 1–7.

[2] J. Davies, D. Fensel, and F. Van Harmelen, *Towards the semantic web: ontology-driven knowledge management*. John Wiley & Sons, 2003.

[3] D. Martín-Lammerding. (2021). Dronetology-cas, the anti-collision ontology, https://dronetology.net/dronetology-cas, [Online]. Available: https://dronetology.net/dronetology-cas (visited on 02/02/2021).

[4] W. W. Group. (2021). SSN, Semantic Sensor Network Ontology, https://www.w3.org/tr/vocab-ssn/, [Online]. Available: https://www.w3.org/TR/vocab-ssn/ (visited on 02/02/2021).

[5] R. M. Keller, S. Ranjan, M. Y. Wei, and M. M. Eshow, "Semantic representation and scale-up of integrated air traffic management data," in *Proceedings of the International Workshop on Semantic Big Data*, 2016, pp. 1–6.

[6] H. Rahman and M. I. Hussain, "A light-weight dynamic ontology for internet of things using machine learning technique," *ICT Express*, 2020.

[7] M. P. Owen, A. Panken, R. Moss, L. Alvarez, and C. Leeper, "Acas xu: Integrated collision avoidance and detect and avoid capability for uas," in *2019 IEEE/AIAA 38th Digital Avionics Systems Conference (DASC)*, IEEE, 2019, pp. 1–10.

[8] C. Muñoz, A. Narkawicz, G. Hagen, J. Upchurch, A. Dutle, M. Consiglio, and J. Chamberlain, "Daidalus: Detect and avoid alerting logic for unmanned systems," in *2015 IEEE/AIAA 34th Digital Avionics Systems Conference (DASC)*, IEEE, 2015, 5A1–1.

[9] J. T. Davies and M. G. Wu, "Comparative analysis of acas-xu and daidalus detect-and-avoid systems," *National Aeronautics and Space Administration NASA Ames Research Center; Moffett Field CA United States Technical Report NASA/TM-2018-219773 ARC-E-DAA-TN50499*, 2018.

[10] Redcat Holdings. (2021). Drone Box, https://www.redcatholdings.com/drone-box, [Online]. Available: https://www.redcatholdings.com/drone-box (visited on 02/02/2021).

[11] Tl-Elektronic. (2021). Black box, https://www.tl-elektronic.com/, [Online]. Available: https://www.tl-elektronic.com/index.php?page=uav&p_id=40&lang=en (visited on 02/02/2021).

[12] UAV Navigation. (2021). Black Box https://www.uavnavigation.com/, [Online]. Available: https://www.uavnavigation.com/sites/default/files/docs/2021-03/UAV%20Navigation%20FDR01%20Brochure.pdf (visited on 02/02/2021).

[13] C. Rekkas and M. Rees, "Towards ads-b implementation in europe," in *2008 Tyrrhenian International Workshop on Digital Communications-Enhanced Surveillance of Aircraft and Vehicles*, IEEE, 2008, pp. 1–4.

[14] D. Zuehlke, N. Prabhakar, M. Clark, T. Henderson, and R. J. Prazenica, "Vision-based object detection and proportional navigation for uas collision avoidance," in *AIAA Scitech 2019 Forum*, 2019, p. 0960.

[15] U. Papa, G. Ariante, and G. Del Core, "Uas aided landing and obstacle detection through lidar-sonar data," in *2018 5th IEEE International Workshop on Metrology for AeroSpace (MetroAeroSpace)*, IEEE, 2018, pp. 478–483.

[16] U. Papa, "Sonar sensor model for safe landing and obstacle detection," in *Embedded Platforms for UAS Landing Path and Obstacle Detection*, Springer, 2018, pp. 13–28.

[17] N. Gellerman, M. Mullins, K. Foerster, and N. Kaabouch, "Integration of a radar sensor into a sense-and-avoid payload for small uas," in *2018 IEEE Aerospace Conference*, IEEE, 2018, pp. 1–9.

[18] A. Muraru, "A critical analysis of sense and avoid technologies for modern uavs," *Advances in Mechanical Engineering ISSN: 2160-0619*, vol. 2, Mar. 2012. DOI: 10.5729/ame.vol2.issue1.23.

[19] D. Martín-Lammerding. (2021). Dronetology, the UAS Ontology, https://dronetology.net/dronetology, [Online]. Available: https://dronetology.net/dronetology (visited on 02/02/2021).

[20] Web Working Group. (2021). SPARQL Query Language for RDF, https://www.w3.org/2001/sw/wiki/sparql, [Online]. Available: https://www.w3.org/TR/sparql11-query/ (visited on 02/02/2021).

[21] ——, (2021). Spin Working Group, Rules for SPARQL, https://www.w3.org/submission/spin-sparql/, [Online]. Available: https://www.w3.org/Submission/spin-sparql/ (visited on 02/02/2021).

[22] ——, (2021). Web Ontology Language (OWL), https://www.w3.org/owl/, [Online]. Available: https://www.w3.org/owl/ (visited on 02/02/2021).

[23] ——, (2021). Resource Description Framework (RDF), https://www.w3.org/2001/sw/wiki/rdf, [Online]. Available: https://www.w3.org/rdf/ (visited on 02/02/2021).

[24] Raspberry Pi Foundation. (2021). Raspberry Pi 3, https://www.raspberrypi.org/, [Online]. Available: https://www.raspberrypi.org/ (visited on 02/02/2021).

[25] D. Martín-Lammerding. (2021). Competency questions in sparql, https://dronetology.net/sim/competency-questions.zip, [Online]. Available: https://dronetology.net/sim/competency-questions.zip (visited on 08/02/2021).

[26] S. Graham, J. De Luca, W.-z. Chen, J. Kay, M. Deschenes, N. Weingarten, V. Raska, and X. Lee, "Multiple intruder autonomous avoidance flight test," in *Infotech@ Aerospace 2011*, 2011, p. 1420.

 ISBN: 978-1-61208-888-4