Ontologies for Cloud Robotics

EDISON PIGNATON DE FREITAS
University Federal of Rio Grande do Sul, Brazil
E-mail: epfreitas@inf.ufrgs.br

JULITA BERMEJO-ALONSO
Universidad Politecnica de Madrid, Spain
E-mail: julia.bermejo@upm.es

ALAA KHAMIS
General Motors, Canada
E-mail: alaakhamis@gmail.com

HOWARD LI
University of New Brunswick, Canada
E-mail: howard@unb.ca

JOANNA ISABELLE OLSZEWSKA
University of West of Scotland, UK
E-mail: joanna.olszew ska@ieee.org

Abstract

Cloud Robotics (CR) is currently a growing area in the Robotic community. Indeed, the use of cloud computing to share data and resources of distributed robotic systems leads to the design and development of Cloud Robotic Systems (CRS) which constitute useful technologies for a wide range of applications such as smart manufacturing, aid and rescue missions, etc. However, in order to get coherent agent-to-cloud communications and efficient agent-to-agent collaboration within these CRS, there is a need to formalize the knowledge representation in Cloud Robotic. Hence, the use of ontologies provides a mean to define formal concepts and their relations in an interoperable way. This paper presents standard robotic ontologies and their extension in the CR domain as well as their possible implementations in the case of a real-world CR scenario.

1 Introduction

The term ‘Cloud Robotics’ (CR) was first coined by Kuffner (2010). It refers to any robot or automation system that utilizes the cloud infrastructure for either data or code of its execution, i.e. a system where all sensing, computation, and memory are not integrated into a single standalone system (Kehoe et al., 2015). Cloud Robotic Systems (CRS) appeared at the beginning of this decade, and their presence is increasing in both industrial utilization (Rahimi et al., 2017) and daily life use (Sharath et al., 2018). Hence, applications of CRS range from smart factory (Rahman et al., 2019) including teleoperation (Salmeron-Garcia et al., 2015) to smart city (Ermacora et al., 2013), encompassing surveillance (Bruckner et al., 2012), urban planning (Bruckner et al., 2012), disaster management
(Jangid and Sharma, 2016), ambient assisted living (AAL) (Quintas et al., 2011), life support (Kamei et al., 2012) and elderly care (Kamei et al., 2017).

Although cloud robotics is still a nascent technology (Jordan et al., 2013), a number of start-ups and spin-offs have been recently created to develop and commercialize cloud robotics targeting different application domains. For example, Rapyuta Robotics (Hunziker et al., 2013), (Mohanarajah et al., 2015), which is an ETH spin-off, builds cloud-connected, low-cost multi-robot system for security and inspection based on RoboEarth (Waibel et al., 2011), (Riazuelo et al., 2015). CloudMinds is a cloud robotics start-up for housekeeping and family services. SoftBank, which is a Tokyo-based phone and Internet service provider, has recently invested $100 million in the cloud robot Pepper created by Aldebaran Robotics. In June 2015, SoftBank Robotics announced that the first 1,000 units of this robot were sold out in less than one minute (Guizzo, 2011).

In particular, CRS are characterized by three levels of implementation (Zhang and Zhang, 2019): (i) the Infrastructure as a Service (IaaS) (Du et al., 2011) to run, e.g. the operating systems on the Cloud; (ii) the platform as a Service (PaaS) to share, e.g. data (Tenorth et al., 2013), system architectures (Miratabzadeh et al., 2016), etc.; and (iii) the Software as a Service (SaaS) (Arumugam et al., 2010) to allow, e.g. the process and the communication of the data. Hence, CRS use the Cloud for robotic systems to share their data about the context (Huang et al., 2019), the environment (Wang et al., 2015), the resources (Chen et al., 2018), their formation (Turnbull and Samanta, 2013), their navigation (Hu et al., 2012), or to perform tasks such as the simultaneous localization and mapping (SLAM) (Benavidez et al., 2015), the grasping of objects (Zhang et al., 2019), leading to define path planning (Lam and Lam, 2014) or robot control as services (Vick et al., 2015), and more generally, Robotic as a Service (RaaS) (Doriya et al., 2012).

Thence, by studying the context of the Cloud Robotics (CR) field, its importance, and the need for standards that can support its growth, this paper brings a contribution in identifying and testing ontologies that aim to define and facilitate access to various types of knowledge representation and shared resources for the Cloud Robotic Systems (CRS).

The rest of the paper is structured as follows. Section 2 presents existing standard ontologies for the Cloud Robotic domain, along with a literature overview about relevant efforts in the CRS field. Section 3 describes a real-world case study providing an insight in the use of the presented ontologies to enhance information sharing in a Cloud Robotic System, while Section VI concludes the work with reflections and future directions.

2 Ontologies for Cloud Robotic Systems

2.1 Overview

In light of the need for standardization of the shared data among the CRS’ agents, the use of ontologies plays an important role (Kitamura and Mizoguchi, 2003). Indeed, ontology specify both formally and semantically the key concepts, properties, relationships, and axioms of a given domain (Olszewska and Allison, 2018). In general, ontologies make the relevant knowledge about a domain explicit in a computer-interpretable format, allowing reasoning about that knowledge to infer new information. Some of the benefits of using ontology are thus the definition of a standard set of domain concepts along with their attributes and inter-relations, and the possibility for knowledge capture and reuse, facilitating systems specification, design and integration as well as accelerating research in the field (Olszewska et al., 2020).

Over the past, there have been several projects focusing on the use of ontologies for cloud robotic applications, in order to express the vocabulary and knowledge acquired by robots in specific context such as bioinformatics (Soldatova et al., 2006), rehabilitation (Dogmus et al., 2015), or kit building (Balakirsky et al., 2012). Indeed, through ontologies, robots are able to use a common vocabulary to represent their knowledge in a structured, unambiguous form and perform reasoning over these data (Muhayyuddin and Rosell, 2015). However, the growing complexity of behaviors that robots are expected to present naturally entails, on one hand, the
use of increasingly complex knowledge, calling for an ontological approach and, on the other hand, the need for multi-robot coordination and collaboration, requiring a standardized approach.

Thence, the remaining sections (Sections 2.2 and 2.3) are focused on two ontologies, namely, the Ontology for Autonomous Systems (OASys) and the Robotic Cloud Ontology (ROCO), which are part of the IEEE standardization effort dedicated to Robot Task Representation (P1872.1) and the Standard for Autonomous Robotics (P1872.2), respectively. As explained in Section 3, these OASys and ROCO standard ontologies can be extended to the Cloud Robotic domain and applied in Cloud Robotic Systems.

2.2 OASys Ontology

The Ontology for Autonomous Systems (OASys) (Bermejo-Alonso and Sanz, 2011) describes the domain of autonomous systems as software and semantic support for the conceptual modelling of autonomous system’s description (ASys Ontology) and engineering (ASys Engineering Ontology) (Fig. 1) and could be used for Cloud Robotics. Indeed, OASys organizes its content in the form of different subontologies and packages. Subontologies address the intended use for autonomous systems’ description and engineering at different levels of abstraction, whereas packages within a subontology gather aspect-related ontological elements. Both subontologies and packages have been designed to allow future extensions and updates to OASys. The different subontologies and packages have been formalized using Unified Modeling Language (UML) (Olszewska et al., 2014).

The ASys Ontology gathers the ontological constructs necessary to describe and characterize an autonomous system at different levels of abstraction, in two subontologies, as follows:

1. The System Subontology contains the elements necessary to define the structural and behavioral features of any system.
2. The AsysSubontology specializes the previous concepts for autonomous systems, consisting of several packages to address their structure, behavior, and function.

The ASys Engineering Ontology is dedicated to system’s engineering concepts. It comprises two subontologies to address at a different level of abstraction the ontological elements to describe the autonomous system’s engineering development process, as follows:

![OASys ontologies, subontologies and packages.](image-url)
1. The System Engineering Subontology gathers the concepts related to an engineering process as general as possible, based on different metamodels, specifications, and glossaries used for software-based developments.

2. The ASys Engineering Subontology contains the specialisation and additional ontological elements to describe an autonomous system’s generic engineering process, organised in different packages.

The OASys-driven Engineering Methodology (ODEM) guides the application of the OASys elements for the semantic modelling of an autonomous system considering the phases, the tasks, and the work products (as conceptual models and additional documents). It uses as guideline the ontological elements in the System Engineering and ASys Engineering Subontologies. ODEM also specifies the OASys packages to be used in each of the phases of the engineering process (Bermejo-Alonso and Sanz, 2011).

2.3 ROCO Ontology

The Robotic Cloud Ontology (ROCO) is based on the Robot Architecture Ontology (ROA) (Olszewska et al., 2017). The ROA ontology, developed by the IEEE Robotics and Automation Society (RAS) Autonomous Robotics (AuR) Study Group (Fiorini et al., 2017) as part of the IEEE-RAS ontology standard for autonomous robotics (Bermejo-Alonso et al., 2018), defines the main concepts and relations regarding robot architecture for autonomous systems and inherits from SUMO/CORA ontologies (Prestes et al., 2013). In particular, essential concepts for effective intra-robot communication (Doriya et al., 2015) such as behavior, function, goal, and task have been defined in ROA (Olszewska et al., 2017). Thence, ROCO ontological framework intends to apply the ROA ontology to Cloud Robotic Systems by extending ROA concepts to the cloud Robotic domain (Fig. 2), in order to enhance information sharing in cloud robotic systems.

![Figure 2: Excerpt of the Cloud Robotic Ontology concepts implemented in OWL language.](http://www.semanticweb.org/journals/ontologies/2017/ROCO.owl)
The ROCO ontology has been implemented in the Web Ontology Language (OWL) (W3C OWL Working Group, 2009), which is well-established for knowledge-based applications (Olszewska et al., 2017), (Beetz et al., 2018), using the Protege tool and following the Robotic ontological standard development life cycle (ROSADev) methodological approach (Olszewska et al., 2018).

3 Cloud Robotic Case Study

3.1 Cloud Robotic Context

The studied system is an interactive, heterogeneous, decentralized multi-robot system (Calzado et al., 2018) that is composed of a set of unmanned aerial vehicles (UAVs) and unmanned ground vehicles (UGVs) that belong to different organizations, such as Regional Police, Federal Police, Firefighters, Civilian Defense and Metropolitan Guard.

These organizations have their own UAVs and/or UGVs to support them individually in different application scenarios. These robots can share data inside their organizations through private cloud services. The robots used by these different organizations could have different characteristics, depending on the organizations’ needs. For instance, multi-rotor UAVs, such as quadcopters or hexacopters, can offer Metropolitan Guard surveillance of specific buildings or public areas, while UGVs from the Federal Police can access places, which are potentially under threat by terrorist attacks, in order to disarm bombs.

The convergence of efforts from different law enforcement organizations can better assist the population against the threats posed by criminals and outlaws. The previously mentioned UAVs and UGVs can form a robotic cloud (Mell and Grance, 2011) to share data about the environment under concern, to consume data, to use resources to perform computation intensive tasks, and to offer their resources to other members of this cloud. As different organizations are sharing the same robotic cloud, it is possible to state that this is a Cloud Robotic System (CRS).

A service that the cloud can provide to the UAVs and UGVs composing this CRS is motion planning. As stated above, there are different types of UAVs and UGVs (e.g. fixed-wing, multi-rotors of different types, aerostats, ground robots with or without manipulators, etc.). Depending on the environment data that are uploaded to the cloud, different forms of event assistance services are needed in different areas. Whatever the required actions to be performed, obstacles may hinder the usage of given UAV or UGV to perform a given task, or make one more appropriate than another. Upon the decision about which UAV or UGV should perform a given task, there is a need to execute its motion planning, in order the robot can move to the area in which it is needed. Moreover, it is also important to specify the tasks which have to be performed (Bruckner et al., 2012).

For this heterogeneous system composed of different types of UAVs and UGVs with different capabilities, different sensors, and different actuators, semantic coherent definitions of the involved concepts have to be provided in order to ensure efficient communication in between the agents (Olszewska, 2017). Indeed, the semantic coherence of these definitions is important so that the required tasks can be assigned to any UAV or UGV that may be able to perform them, regardless it specific type. In particular, the cloud-robotic-based system, which owns the need for clear semantic processing of tasks allocation in context of this surveillance application (as described in Section 3.2), can benefit from a standard ontology (as illustrated in Sections 3.3-3.4).

3.2 Cloud Robotic Scenario

As a real-world example of this overall CRC system architecture, we consider that 2 UGVs (i.e. a wheeled robot with a surveillance camera and a gripper (UGV-1), and a wheeled robot with only a surveillance camera (UGV-2)) are used for surveillance by Neighborhood Patrol Division of the Metropolitan Guard, and that 2 UAVs (i.e, a conventional helicopter (UAV-1) and a quadrotor (UAV-2)) are used for patrolling operations by the Highway Patrol Division of the Regional
Figure 3: Cloud Robotic Scenario Environment with Unmanned Aerial Vehicles (UAVs) and Unmanned Ground Vehicles (UGVs).

Police. All these vehicles use cloud computing facilities to process data and/or computation intensive algorithms such as SLAM/cooperative SLAM, sampling-based planning or planning under uncertainty, since it is much faster than relying on their own onboard computers.

On the other hand, all these UGVs and UAVs evolve in an environment, as illustrated in Fig. 3, i.e. with 9 Volumes of Interest (VOIs). These VOIs consist of 6 Areas of Interest (AOIs), 2 High Ways (HWs), and a base station (BS). In particular, AOI-1 is a rural area, AOI-2 is an industrial zone, AOI-3 is a sport complex, AOI-4 is a residential area, AOI-5 is a service area, and AOI-6 represents a gas station and its surroundings.

To perform daily routine surveillance and patrolling operations, it is required in the scenario to deploy these vehicles starting from the base station of the Regional Police and then to bring them back to this base station. It is assumed that the VOI of each vehicle is pre-assigned as follows: AOI-1, AOI-6, and HW-1 are assigned to UAV-1; AOI-4, AOI-5, and HW-2 are assigned to UAV-2; AOI-2 is assigned to UGV-1; and AOI-3 is assigned to UGV-2.

Part of this deployment mission is to plan the motion of each vehicle from the BS to a certain vantage point in the pre-assigned VOI and vice-versa. In this planning problem, the goal is to determine a future course of actions/activities for an executing entity (UAV/UGV) to drive it from an initial state (BS) to a specified goal state (VOI). It is supposed that at the beginning of the scenario, all UAVs and UGVs are in the base station, and all AOIs and HWs are not under surveillance/patrolling. Besides, it is assumed that the CRS system must be able to provide ground robots with a safe manner to reach their destination AOIs, as they are not able to have the same overview as the UAVs. Indeed, UGV-1 has to go through AOI-5 and HW-2 to reach AOI-2, while UGV-2 has to go through AOI-4 and HW-2 to reach AOI-3.

Considering the cooperative manner a cloud-robotic system operates, UGV-1 and UGV-2 can request to the cloud a service for clearance about these places that they have to go through, in order they are able to make a safe move towards their destination AOIs. Thence, this request for clearance triggers a query to all robots connected to the cloud, and thus, a request capability
match takes place. This matching has to consider both the necessary capabilities related to surveillance areas and the situation or context. In this case, it consists in matches between what the robots are able to do and about where the information is required (i.e. AOI-4, AOI-5, and HW-2).

According to what is stated in the presented scenario, UAV-2 has the necessary capabilities to provide the requested information, and for this, it must behave in a way to perform the actions of moving to AOI-4, collecting an image of AOI-4, then moving to AOI-5, capturing an image of this area and next, moving to HW-2, and collecting an image of this place. Afterwards, it has to share within the cloud structure the requested data which will be used by the robots that demanded the clearance over the areas.

3.3 OASys Ontology Implementation

The case study presented in Section 3.2 is analyzed in this subsection focusing on the application of the proposed OASys ontological framework (Section 2.2) in a cloud-based robotic system that has the goal to provide law enforcement services (Section 3.1).

It is worth noting that ontological terms are written in this section as usual to ease the reading process, whereas they are named as a unique word capitalizing each word within it as made in the naming convention in OASys. Italics are used for the Case Study scenario components specified in Section 3.2.

Given the fact that the proposed case study is a Multi-agent System (MAS) scenario, it is possible to use MAPL (Multi-Agent Planning Language) or MA-PDDL (Multi Agent PDDL) as extensions of the Planning Domain Definition Language (PDDL) (McDermott et al., 1998) in multi-agent scenarios. MA-PDDL (Kovacs, 2012) provides the ability to represent different actions of different agents or different capabilities, different goals of the agents and actions with interacting effects between the agents such as cooperation, joint-actions and constructive synergy.

For sake of simplicity, it is assumed that there are no cooperative behaviors/joint actions between the vehicles in the proposed scenario, and then, PDDL can be used. Objects, Predicates, Initial state, Goal specification and Actions/Operators are the main components of a PDDL planning task. Indeed, objects represent things in the world that interest us. Predicates are properties of objects that are of interest to be studied; they can be true or false. Initial state is the state of the world that the system starts in. Goal specification represents things that is desirable to be true; and finally actions/operators are ways of changing the state of the world or capabilities.

In the proposed scenario, it is possible to extract the following examples of these five components:

1. **Objects**: The 2 UAVs, the 2 UGVs, the 9 VOI. (At lower level of abstraction for each vehicle, it is possible to consider planning entity, planned/execution entity, monitoring entity, motion planning solver, and planning environments as objects.)
3. **Initial state**: All UAVs and UGVs are in the base station. All AOI and HWs are not under surveillance/patrolling.
4. **Goal specification**: UAV-1 is in AOI-1/AOI-6/HW-1; UAV-2 is in AOI-2/AOI-3/HW-2; AGV-1 is within AOI-4 and AGV-2 is within AOI-5.
5. **Actions and Operators**: Each vehicle can execute same/different motion plan shared via the cloud to move from the base station to the AOI/HWs and vice-versa. It can avoid stationary and/or moving obstacles while executing the plan, or it can continuously monitor the execution of the plan and it can adapt/repair the plan to resolve conflicts, if any.

For example, we assume the plan execution case as follows:
1. **Objects:** VOI: AOI-1, AOI-2, AOI-3, AOI-4, AOI-5, AOI-6, HW-1, HW-2, and Base-Station Vehicles: UGV-1, UGV-2, UAV-1, UAV-2

   **In PDDL:**
   
   AOI-6,
   1) (:objects AOI-1, AOI-2, AOI-3, AOI-4, AOI-5, AOI-6,
   2) UGV-1, UGV-2, UAV-1, UAV-2)

2. **Predicates:**

   VOI(x) true iff x is a AOI/HW

   Vehicle(x) true iff x is a UGV/UAV

   at-Base-Veh(x) true iff x is a Base-Station and a Vehicle is in x

   at-VOI-Vehicle(x,y) true iff x is a VOI and VOI Base-Station and a Vehicle y is in x

   **In PDDL:**

   (:predicates (VOI ?x) (Vehicle ?x)
   (at-Base-Veh ?x) (at-AOI- UGV ?x)
   (at-AOI- UAV ?x) (at-HW- UAV ?x))

3. **Initial state:**

   VOI(Base-Station) is true. Vehicle(UGV-1), Vehicle(UGV-2), Vehicle(UAV-1), Vehicle(UAV-2) are true. at-Base-Veh(UGV-1), at-Base-Veh(UAV-1), at-Base-Veh(UAV-2) are true. Everything else is false.

   **In PDDL:**

   (:init (VOI Base-Station) (VOI AOI-1) (VOI AOI-2) (VOI AOI-3) (VOI AOI-4) (VOI AOI-5) (VOI AOI-6) (VOI HW-1) (VOI HW-2) (Vehicle UGV-1), (Vehicle UGV-2), (Vehicle UAV-1), (Vehicle UAV-2) (at-Base-Veh UGV-1), (at-Base-Veh UAV-2), (at-Base-Veh UAV-2))

4. **Goal specification:**


   **In PDDL:**

   (:goal (and (at-VOI-Vehicle AOI-1 UAV-1)
   (at-VOI-Vehicle AOI-1 UAV-1)
   (at-VOI-Vehicle AOI-1 UAV-1)
   (at-VOI-Vehicle AOI-6 UAV-1)
   (at-VOI-Vehicle HW-1 UAV-1)
   (at-VOI-Vehicle AOI-2 UAV-2)
   (at-VOI-Vehicle AOI-3 UAV-2)
   (at-VOI-Vehicle HW-1 UAV-2)
   (at-VOI-Vehicle AOI-4 UGV-1)
   (at-VOI-Vehicle AOI-5 UGV-2)))

5. **Action and Operators:**

   **Description:** The vehicle can execute motion plan available on the cloud to move from the base station and between the AOI/HWs, can avoid stationary and/or moving obstacles, can monitor the execution of the plan, and can adapt/repair the plan to resolve conflicts. For sake of simplicity, assume that a vehicle y moves from a to b.

   **Precondition:** VOI(a), VOI(b) and at-VOI-Vehicle(a,y) are true.

   **Effect:** at-VOI-Vehicle(b,y) becomes true. at-VOI-Vehicle(a,y) becomes false. Everything
Ontologies for Cloud Robotics

else does not change.

**In PDDL:**

```
(:action move :parameters (?a ?b)
 :precondition (and (VOI ?a) (VOI ?b)
 (at-VOI-Vehicle ?a))
 :effect (and (at-VOI-Vehicle ?b)
 (not (at-VOI-Vehicle ?a))))
```

Considering the OASys-driven ODEM, its Analysis Phase involves different tasks performed for the Structural, Behavioral and Functional Analysis of the system. As such, the Structural Analysis task models the system structure and knowledge (Fig. 4). The System Modelling subtask models the structural elements in the system and their relationships, obtaining a Structural Model, based on the ontological elements in the System Subontology. OASys allows to model the abstract knowledge to be used in the system. Later on, the motion planning process uses the instantiated knowledge for the particular system under specification.

In this sense, Fig. 5 shows the Structure Model specialising OASys System Subontology abstract knowledge classes (System, Subsystem, Element) and their composition relationships using a UML class diagram notation. The lines distinguish the different level of abstraction, i.e. any system concepts at the upper level, the CRC system concepts at the intermediate level as specialization through isA relationships, and the actual Objects in the Case Study as instances of the previous classes through instanceOf relationships (not all shown to avoid cluttering the diagram). This Structure Model is complemented with the Topology Models to show the topological and mereological relationships. These models allow representing the Predicates specifying which specific Vehicle surveys, patrols or is within the limits of a concrete VOI.

The role played by this ontological-based Structural Model is twofold. Firstly, it allows the different elements in the CRC System to know about the different, involved vehicles and VOIs and the topological as well as mereological relationships among them for a further reasoning on the motion planning tasks. Secondly, the models can also be used as a semantic system specification for a Model-based Systems Engineering (MBSE) process.

In a similar way, the Knowledge Modelling subtask obtains a Knowledge Model to be used by the system. In this case, the main knowledge to model is the planning related concepts about Initial State and the Goal Specification, in a Goal Structure Model. Figure 6 shows how the abstract knowledge is conceptualized in OASys. As illustrated in this figure, concepts such as Goal, RootGoal, SubGoal and LocalGoal are specialized for the CRC System at the intermediate level, while UML classes are used to specify generic Vehicle Goals. Later on, the actual objects for the Case Study become instances of the former classes, as shown in the UML object diagram, with attributes and operations to define the case study’s specific goals to be fulfilled by the Vehicle (i.e. AGVs and UAVs) in relation to the VOI (i.e. AOI, HW, and BS). This semantic model constitutes the first kind of knowledge for the CRC system to make decisions on during the motion planning reasoning. Other conceptual models using ODEM (Bermejo-Alonso et al., 2016) can be obtained in a similar way in order to specify the system behavior in terms of the motion planning Actions (e.g. avoid obstacles, resolve conflicts, etc.), as shown in different UML activity and sequence diagrams by specializing OASys ontological concepts and relationships.

Hence, the first qualitative results of OASys have been presented in Figs. 4-6 in context of this CRS surveillance application. As a first attempt to characterise the CRS domain, it provides enough support to conceptualise this case study. Currently, OASys is under revision to be implemented using OWL and reasoning mechanism to enhance both the characterisation of the robotic systems (Balakirsky et al., 2017) and the metacontrol features (Hernandez et al., 2018). In particular, OASys is considered for the standardisation efforts under development by the IEEE P1872.1 working group focused on Robot Task Representation (RTR).
Figure 4: Structural analysis task defined in ASys engineering subontology.

Figure 5: Structure model specializing concepts for the case study.
3.4 ROCO Ontology Implementation

The ROCO ontological framework introduced in Section 2.3 represents robots' actions, functions, behaviors, and capabilities using the standard OWL language as demonstrated in context of the scenario described in Section 3.2.

The actual objects present in the case study are instances of the proposed CRS ontological concepts and are related by the defined axioms.

Particularly, the agents UGV1 (Fig. 7) and UGV2 (Fig. 8), which are both requiring clearance from UAV2 (Fig. 9), are considered, since they navigate through the same VOIs; all these agents are being part of the same CRS, as implemented in OWL (Fig. 10) and displayed in Fig. 11. Therefore, UGV1 and UGV2 could use the CRS as a resource to request clearance. This has been implemented in OWL by means of the `isRequestingClearance` property, as illustrated in Fig. 12.

Hence, the first qualitative results of ROCO have been presented in Figs. 7-15 in context of this CRS surveillance application. To sum up, the role played by this ontological-based cloud robotic model is twofold. Firstly, it allows the different elements in the CRS to know which agents (i.e. vehicles) are involved in and what are their positions (i.e. VOIs) as well as the
Figure 7: Snapshot of the UGV1 instance of Agent. Yellow rows are automatically inferred by the reasoner. Best viewed in colour.

Figure 8: Snapshot of the UGV2 instance of Agent. Yellow rows are automatically inferred by the reasoner. Best viewed in colour.

topological and mereological relationships among them, leading to further reasoning on their motion planning (Figs. 13-15). On the other hand, this approach is useful to establish a common shared vocabulary to enhanced CRS agents’ communication. Thereupon, ROCO is considered for the standardisation efforts under development by the IEEE P1872.2 working group focused on Autonomous Robotics (AuR).
Figure 9: Snapshot of the UAV2 instance of Agent. Yellow rows are automatically inferred by the reasoner. Best viewed in colour.

Figure 10: Snapshot of the CRS instances implemented in OWL.

Figure 11: Snapshot of the CRS instance of CloudRobotic System. Yellow rows are automatically inferred by the reasoner. Best viewed in colour.
Figure 12: Snapshot of the `isRequestingClearance` property implemented in OWL.

Figure 13: Snapshot of the reasoning on the `Action` concept in context of the CRS case study.

Figure 14: Snapshot of the reasoning on the `Capability` concept in context of the CRS case study.
3.5 Discussion

OASys is an ontology that comprises both autonomous systems description and its engineering process as mentioned in Section 2.2. The upper level regarding a general system is well founded, based on former standard and proved theories. Thence, the autonomous system level, both from a description and an engineering view would benefit of the current standardisation efforts by IEEE, e.g. in its P1872.1 Task Representation Ontology project. Indeed, the Action package in the ASys Subontology of OASys is an initial attempt to characterise goals, tasks, and functions in autonomous systems, which are aligned with the focus of the IEEE P1872.1 under-progress standard and which are useful for cloud robotic systems.

The ROCO ontology in turn follows the METHONTOLOGY ontological methodology approach to characterise autonomous systems, and in particular to contribute to the IEEE P1872.2 ontology under development by the Autonomous Systems Working Group, as pointed out in Section 2.3. Thus, ROCO inherits from the autonomous robot architecture ontology which was developed in the context of the IEEE P1782.2 Autonomous Robotic effort, while it has a greater focus on cloud robotics. Indeed, ROCO considers concepts related to behavior, function, task, etc. in order to specifically formalized the cloud robotics domain.

Hence, both ontologies contribute towards the current standardisation efforts by the IEEE, regarding robotics and autonomous systems, and in particular CRS specific concepts and relations. It is worth noting that OASys was developed using a systems engineering language called UML, which is considered to be a lightweight language without an embedded reasoning mechanism. Consequently, the reasoning on OASys has to be processed externally, e.g. on a PDDL platform as explained in Section 3.3. Unlike OASys, the ROCO ontology has been directly developed in OWL, which is a computational logic-based language such that knowledge expressed in OWL can be exploited by computer programs and automated reasoners, e.g. to verify the consistency of that knowledge or to make implicit knowledge explicit, as described in Section 3.4.

Therefore, OASys ontological engineering has served as a general systems approach and a common taxonomic basis for the IEEE P1872 ontologies in development, while the ROCO ontology has allowed to specialise the standardised ontologies to include cloud-robotic specific concepts and relations which automated reasoning can be directly performed on.
4 Conclusions

Cloud Robotic Systems (CRS) alloy cloud computing and autonomous multi-robot systems. These technologies require an adequate understanding and a coherent sharing of the resources and data among the CRS agents in order to make possible an interoperable communication and efficient collaboration between the CRS stakeholders.

For this purpose, ontologies have been identified as a possible solution. Thence, this paper presents the current state of standard ontologies for the Cloud Robotic domain. Moreover, a case study showcases the implementation of the described ontologies, namely, the Ontology for Autonomous Systems (OASys) and the Robotic Cloud Ontology (ROCO), into a real-world motion planning CRS scenario, highlighting the main features of these ontologies in the CR context and reporting the results obtained when using the defined ontologies to enhance the interoperability of cloud robotics systems.

Future works aim to enrich the vocabulary developed so far in the existing standard ontologies and create further semantic extensions that will contribute to the ontological standard which is being developed by the IEEE Ontologies for Robotics and Automation (ORA) Working Group and its Robot Task Representation (RTR) and Autonomous Robotics (AuR) subgroups.

References


Ontologies for Cloud Robotics


