Service Oriented Computing and Applications (2023) 17:25–37
https://doi.org/10.1007/s11761-022-00356-2

ORIGINAL RESEARCH PAPER

A core IoT ontology for automation support in edge computing

Sahar Ghrab[1] · Imene Lahyani[2] · Sami Yangui[3] · Mohamed Jmaiel[2]

Received: 28 July 2022 / Revised: 20 December 2022 / Accepted: 31 December 2022 / Published online: 10 February 2023

Abstract
Service providers provision more and more Internet-of-Things (IoT) services in the cloud for dynamicity and cost-effectiveness purposes. This is made possible thanks to the introduction of edge computing that brings additional computing and resources for analytics close to the data sources and thus enables meeting the low latency requirement. Edge nodes should support (i) the heterogeneity of IoT devices (e.g., sensor, actuator) and (ii) characteristics (e.g., mobility, location awareness). IoT is already integrated to the hybrid cloud/edge environment. However, the ecosystem lacks of automation due to the previously mentioned characteristics. Indeed, edge nodes are often manually selected during deployment time, and most of the regular Quality Of Service (QoS) management procedures remain difficult to implement. This paper introduces a comprehensive semantic model called EdgeOnto. It encompasses all concepts related to IoT applied in the context of edge computing. The ultimate goal of EdgeOnto is to automate the several steps that make up the IoT services lifecycle in hybrid cloud/edge environment. On the one hand, semantics enable an automatic discovery of the relevant edge nodes that are suitable to host and execute IoT services considering their requirements. On the other hand, it allows supporting the specific QoS procedures that are related to such setting (e.g., low latency, mobility, jitter). The core ontology was designed with the Protégé open-source tool. A smart strawberry farming use case was implemented and evaluated for illustration purposes. The results validate the accuracy and the precision of the designed semantic matchmaker.

Keywords Edge computing · Internet of Things (IoT) · Ontology · Quality of Service (QoS) · Semantic matchmaking

1 Introduction

Internet-of-Things (IoT) service owners are more and more leaning on hyperscale cloud providers to provision IoT services, as well as, to store and to process the generated data. This trend is motivated by the several advantages that the cloud computing paradigm brings such as dynamicity, flexibility and cost-effectiveness. However, with the ever-growing workloads that is tied to 5G telco network, automation, real-time analytics and IoT, service owners started looking elsewhere for their computing needs.

Notably, edge computing enables meeting these computing needs. In fact, edge computing is currently reshaping IT and business computing. It brings additional resources close to the end-users and data sources so that the latency-sensitive computing and data analytics could be processed within a low network latency [24]. Nevertheless, this brings additional challenges that need to be tackled to provision IoT services in such setting. Indeed, edge nodes are highly heterogeneous in terms of computing capabilities (e.g., supported run-time, CPU and RAM workload, storage capacity, autonomy), network interfaces (e.g., supported communication protocols), behavior (e.g., mobile or motionless, online or offline). These characteristics make the operation of edge computing difficult and costly [24], in particular for IoT where applications are latency-sensitive and services are often bound through complex gateways and overlay networks.

IoT is already integrated to the hybrid cloud/edge environments. However, the existing works mainly focus on

✉ Sahar Ghrab
ghrab.sahar@gmail.com

Imene Lahyani
imen.lahyani@enis.tn

Sami Yangui
yangui@laas.fr

Mohamed Jmaiel
mohamed.jmaiel@redcad.org

[1] MIRACL Laboratory, University of Sfax, Sfax, Tunisia

[2] REDCAD Laboratory, University of Sfax, Sfax, Tunisia

[3] LAAS-CNRS, Université de Toulouse, INSA, 31400 Toulouse, France

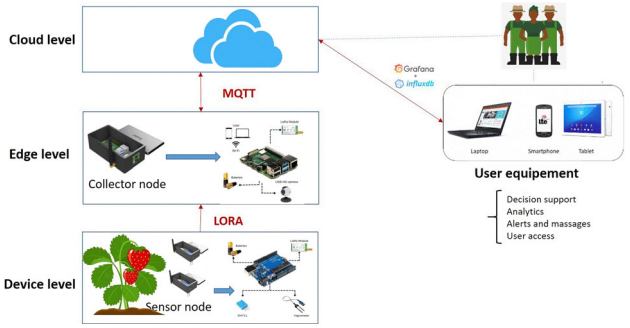

Fig. 1 Smart strawberry farming

specific features (e.g., see [8] for IoT services deployment, see [22] for IoT services rescheduling) and fail to fully support the whole steps that make up the IoT services life cycle. The reader should note that IoT services inherit the same Service-Oriented Architecture (SOA) life-cycle phases (i.e., discover, deploy, execute and manage during run-time) [34]. SOA principles (e.g., service abstraction, discover ability, and composability) ensure the viability of network services' ecosystem that could be dynamically and flexibly provisioned, thus coping with changeable IoT needs and dynamic Quality Of Service (QoS) requirements along with context conditions. In addition to these limitations, the current edge frameworks for IoT lack of automation when implementing these steps. Edge nodes are often manually selected during deployment time, and most of the regular QoS management procedures remain difficult to implement. The QoS procedures need to be executed during run-time and aim to optimize a set of metrics based on predefined service-level objectives. The latter might vary from one use case to another depending on the involved data and IoT devices and expected outcomes. The objectives could, for instance, range from faster response time to less network costs, low latency, less service breakdown or better flow control of data, communication, efficiency and accuracy [10].

1.1 Motivating use case

The motivating use case is from smart agriculture research field. It is inspired from the use case introduced in [12] where the authors propose an edge framework to handle the collection, analysis, prediction, and detection of heterogeneous data in strawberry farming. Strawberries are sensitive fruits that are afflicted by various pests and diseases. Therefore, there is an intense use of agro-chemicals and pesticides during production. Due to their sensitivity, temperatures or humidity at extreme levels can cause various damages to the plantation and to the quality of the fruit.

The proposed framework is depicted in Fig. 1. This framework monitors, collects and processes data in real time to (i) detect seven of the most common strawberry diseases and (ii) determine the required type and amount of chemical fertilizer and medication depending on the strawberry size, age and disease. To that end, it relies on several sensor nodes, a collector node and actuators. These devices communicate with the computing nodes through the collector module that plays the role of IoT gateway. The communication relies on LoRa protocol. The computing nodes integrate machine learning capabilities for capturing outliers in collected data and execute a computer vision model using Yolo v5 architecture to identify the prospective disease.

The provisioning of this framework in large strawberry farms requires, in addition to the deployment of the sensors and actuators, the deployment of several computing entities (e.g., raspberry nodes) that could host the computer vision and machine learning algorithms. These nodes must be sufficient in number to cover the whole farm and to allow close and fast interactions between the devices and the algorithms. The end-to-end selection should optimize the data transfer time and the system reaction (e.g., detecting a disease, control an irrigation actuator for plant watering with medication) as fast as possible and with minimal energy cost to save devices autonomy and durability. Furthermore, one can imagine the deployment of monitoring mobile robots over the farm. The robots will be responsible of holding the monitoring sensors and the actuators instead of having them motionless. In that case, the system should be able to support the devices mobility and dynamically select the most appropriate gateway and computing nodes to optimize on-the-fly the previously mentioned service-level objectives. A precise and relevant semantic description of the nodes capabilities and current workload is the considered alternative in this work to automate and enable efficient selection of the edge nodes while optimizing the QoS metrics.

1.2 Contributions and obtained results

This paper introduces a comprehensive ontology (called EdgeOnto). This ontology involves semantic concepts for both IoT domain and edge domain. It supports edge computing features and integrates high-level abstraction of the complex information and incremented knowledge at the network edge (e.g., ad hoc topology, breakdown, mobility). The ultimate goal is to support the automation of the QoS management procedures in such environment. Specifically, this is achieved by enabling the instantaneous semantic matching between the IoT services requirements, from one side, and the edge nodes workload/position on the other side.

The core domain of EdgeOnto was designed with Protege,[1] a free, open-source ontology editor and framework for building intelligent systems. As for the validation, the smart strawberry farming use case was implemented and evalu-

[1] https://protege.stanford.edu/.

ated. The results validate the accuracy and the precision of the designed semantic matchmaker.

2 The state of the art

This section firstly summarizes the different ontologies proposed in IoT domain. The reviewed ontologies cover sensor, security, and applications ontologies. These ontologies focus on devices (sensor, actuator), its lifecycle, etc. The detailed IoT ontologies do not support the concepts of security, location, place, or time in the exception of some which do not treat it deeply. These important concepts are especially useful in IoT domain in the context of edge computing (see Sect. 1). For thus, some ontologies related to location, context, and time are integrated in this section.

2.1 Ontologies for sensors

Existing ontologies for sensors in the literature aim to solve heterogeneity problems associated with the hardware, software, and the data management aspects of sensors (sensor capabilities, extensibility, data access and sharing, sensor data description, sensor discovery).

The SSN (Semantic Sensor Network) ontology [11] des cribes sensor resources and the data collected through these sensors. The main concepts proposed are sensor, observation, and device.

The SOSA (Sensor, Observation, Sample, and Actuator) ontology [19] is a redesigned SSN ontology. The SAREF (Smart Appliance REFerence) ontology [33] reuses and aligns concepts and relationships in existing appliance-based ontologies and is dedicated to the management of energy and services in smart homes. Its main proposed concepts are Device, Service, Energy, Task, Function and Measurement.

Semantic actuator network (SAN) ontology [26] is used for the description of the actuator concept and its characteristics (Actuator, Actuating Device and Actuation). The OntoSensor ontology [28] extends concepts from other ontologies (SensorML, Sumo, ISO-19115) to allow concepts for the identification of sensor categories, behavior, relationships, functionalities, and meta-data regarding sensor characteristics, performance, and reliability. The MyOntoSens ontology [25] is an extension of OntoSensor. It describes sensor observations and capabilities to reason over the collected data for the domain of wireless sensor networks (WSN).

2.2 Context-aware ontologies

Context-aware ontologies aim to identify information about context like place, agent, event, latitude, longitude, altitude, and location. In the computation context, they express that "*context is any information that can be used to characterize the situation of an entity. An entity is a person, place or object that is considered relevant to the interaction between a user and an application*".

Context can be classified on the one hand as external or internal and the other hand as physical or logical [5]. External or physical contexts are those that can be measured using physical sensors, while internal or logical contexts are those that are explicitly specified by users or captured by monitoring user interactions. Context awareness bridges the gap between real things and the virtual world in the IoT through acquiring, analyzing, and interpreting relevant context information [23]. [27] defines an Internet-of-Thing context-awareness model that accurately represents context in IoT. The context model is a hierarchical structure showing contexts related to Resources, Actors, Ambients and Policies. This context model includes User context, Location context, Activity context, Personal context, Mood context, Time context, System context, Environment context and Device context. Context-aware ontologies are classified into generic ontologies (information provided by mobile device sensors) and specific domain ontologies (university domain, smart information provided by mobile device sensors, both physical (e.g., WiFi, Bluetooth, etc.) and virtual (e.g., user schedule, web-logs, etc.) to support context-aware services. The proposed ontology defines the relations between different user locations and the contexts identified. The ontology proposed by [3] supports the devices's discovery and their location in smart-home domains using concepts like Person, Sensor, Device, Location, etc. The ontology CONON (CONtext ONtology) proposed by [32] models context in pervasive computing environments and supports logic-based context reasoning.

2.3 Location-aware ontologies

Location-aware ontologies describe locations of things like geographical coordinates (altitude, latitude, and longitude). Location is used to describe the context location, its indoor and outdoor space, and the property of the environment (external context). Location is used to describe the spatial context (partly, physical context) of users/devices [4]. Location and context are closely linked and dependent. Different models can be used for defining entities locations [15]. The considered models are as follows:

- Geometric models (comprising Cartesian coordinates)
- Set-theoretic models (for defining location as an element of a set, e.g., cellular location, WiFi AP location, etc.)
- Graph-based models (for defining locations in physically grounded networks, social networks, etc.)
- Semantic models (for defining locations defined using human-friendly notations)

In IoT domain, location informations are often collected from sensor data (like users' mobile devices to estimate device location). WGS84 ontology describes abstract concepts for defining spatial things (buildings, people, etc.) and temporal things (events, or time duration). It also describes the geographical locations of these things by using concepts for defining the geo-coordinates using latitude, longitude, and altitude.

2.4 Time-based ontologies

Time-based ontologies are used to describe the temporal context which include time, duration, and some temporal aspects. Time modeling detects the behavior of a system at an interval or duration or point of time or the actions to be performed in specific temporal entity. The most popular and commonly used time based-ontology is OWL-Time [17]. It is reused to propose other time-based ontologies. It is focused on describing date-time information specified in Gregorian calendar format.

DAML-Time ontology [18] is focused on concepts to provide a common understanding of time, whereas DAML-S ontology [1] provides temporal concepts required to define a web service such as profile, process, and time. The KSL-Time (Knowledge System Laboratory) ontology distinguishes between different types of intervals and granularity. The Timeline ontology [20] extends Time Ontology by providing various concepts representing granularity of time to provide more flexibility to the annotations.

2.5 Security and QoS ontologies

STAC ontology [16] is an IoT security ontology. It defines the main security concepts (cryptographic concepts, security protocols, security tools, security properties) and classifies threats and countermeasures by domain and according to the OSI model. The main purpose of this ontology is to be reused in numerous domains such as security of web applications, network management or communication networks (sensor, cellular and wireless).

In IoT domain, cloud computing and edge computing are widely used to store, deploy and analyze data coming from IoT devices. Due to network overload and changing transmission delay, the QoS offered can be different. That's why, it is important to measure the QoS, which is dependent on many factors like availability, network, robustness, security, scalability, performance, etc. In the literature, different QoS ontologies are proposed. QoS is the description or measurement of the overall performance of a service (like telephony, computer network) particularly the performance seen by the users of the network. Jiang and Aagesen [21] propose a QoS ontology based on functional properties representing service functionality, which are modeled in terms of operations, inputs, outputs, preconditions and effects. However, while non-functional properties comprise business policies, QoS properties as well as context policies. QoS properties include QoS parameters and QoS policies (rule). QoS parameters represent security, availability, scalability, reliability, performance, etc.

[31] proposes an ontology for knowledge representation in the Internet of Things composed of different modules like IoT services, IoT resources, QoS and QoI. In IoT domain, QoS and QoI (Quality of information) are important, which exhibit a much higher level of dynamicity. In this work, [31] enumerates all the QoS parameters and QoI that are common to many applications domains. Reference [31] reuses DUL ontology for ontology building. This ontology will be reused for EdgeOnto building and precisely for the QoS.

2.6 Ontologies for IoT applications

In IoT domain, many ontologies are proposed: IoT ontology [13], IoT Lite [7], and IoT-O [29].

IoT ontology [13] is based on the reuse of other existing ontologies like SSN ontology, DUL (DOLCE Ultralite Upper ontology) ontology and QUDT (Quantities, Units, Dimensions and Data Types) Ontology, whereas IoT-Lite ontology reuses SSN ontology. The description of IoT concepts is based on three classes: objects, system or resource and services. IoT devices are classified into three classes: sensing devices, actuating devices, and tag devices.

IoT-Lite is focused on sensing, although it has a high-level concept on actuation that allows any future extension on this area. Services are described with a coverage representing the 2D-spatial covered by the IoT device.

IoT-O ontology presents connected device networks and semantically describes devices and data in order to make systems aware of their environment, its evolution, and the changes they can bring to it. Such a description allows smart agents to transform their environment thanks to connected actuators, according to the perceptions they have of it through connected sensors. IoT-O is based on following modules:

- Sensing module: describes the input data. Its main classes come from SSN ontology (ssn:Observation, ssn:Sensor, ssn:Device, etc.).
- Acting module: describes how the system can interact with the physical world. Its main classes come from SAN (san:Actuator and san:Actuation). It also reuses SSN classes that are not specific to sensing such as ssn:Device
- Life-cycle module: models state machines to specify system life cycles and device usage. Its main classes are lifecycle: State and lifecycle:Transition.
- Service module: represents web service interfaces. Its main classes come from msm:Service and msm: Opera-

tion. Services produce and consume msm:Messages, and RESTful services can be described with hRest.
- Energy module: IoT-O's energy module is defined by PowerOnt. It provides the poweront:PowerConsumption class, and a set of properties to express power consumption profiles for appliances.

3 Requirements and related work review

The previously discussed ontologies summarize the most important concepts used in IoT domain (sensor, actuator, device, service, etc.). Data generated from the different devices are important and should be stored, treated and analyzed through the nearest edge nodes in order to increase high bandwidth, faster treatment, less transmission delay and less packet loss. That's why different concepts should be tacked into consideration in IoT ontologies proposed. These concepts are related to edge computing characteristics. In fact, edge node should support mobility, data heterogeneity, context-awareness (time and location), security and QoS.

The assessment of existing IoT ontologies regarding the presence of key concepts is summarized in Table 1. These ontologies are evaluated among a set of conceptual requirements detailed below (see paragraph 3).

A set of symbols are used to determine whether the conceptual requirement is supported by the ontology or not:

- *: The conceptual requirement is not supported by the ontology.
- ** : The conceptual requirement is quite supported by the ontology.
- ***: The conceptual requirement is well supported by the ontology.

Requirements The evaluation of related ontologies is based on conceptual requirements identified as follows:

- CR1: "Cloud" constitutes the servers accessible on the Internet as well as software and databases which run on these servers. The servers located in the cloud are hosted in data centers distributed around the world.
- CR2: "Edge" constitutes a computer that acts as an end user portal for communication with other nodes. In this paper, edge node allows the communication between cloud and IoT device.
- CR3: "Location" constitutes context location of an entity in general (e.g., device, actuator, cloud, edge). A distinction is made between physical place and virtual place.
- CR3.1: "Physical place" constitutes an absolute position with geographic coordinates.
- CR3.2: "Virtual place" constitutes a relative position of an entity.
- CR4: "Time" constitutes the information's given for time description (point in time, interval, duration, etc.).
- CR5: "Security" constitutes a characteristic of using data. Data can be accessed only to suitable persons in suitable time and location and with some privilege.
- CR6: "QoS" constitutes the quality of service offered in terms of properties and metrics used.

None of the above-reviewed work meet all the requirements (Table 1). More specifically, some ontologies meet partially the third, fourth, fifth and seventh requirements (time and context-awareness). None meets even the first, the second and the third requirements (edge, cloud, virtual location).

Nowadays, the use of IoT devices is coupled by the use of cloud computing and edge computing for many reasons.

First, it is an opportunity to analyze data generated by IoT devices locally (the nearest edge node used) without sending it to cloud. This can facilitate useful data identification and reduce packet loss and transmission delay. Some applications need real-time treatment, which can be guaranteed by edge computing.

Second, edge node support data heterogeneity of IoT devices.

Finally, it is an opportunity to know the localization of an object which can be physical or virtual, the quality of service offered and information's about time.

To put it in a nut shell, an Iot ontology based on the use of edge computing should integrate other concepts useful in addition to the concepts already presented in the literature in relation to device, life cycle, consumption and other.

The next section gives an overview about these concepts and the different requirement which should be respected on the one hand. On the other hand, it details reused ontologies for the design process of our EdgeOnto ontology.

4 EdgeOnto principles and design model

The design process of EdgeOnto is based on the NeOn methodology presented in [14]. The NeOn methodology is used for building ontology networks and is based on the "divide-and-conquer" strategy which decomposes the general problem to be solved in different sub-problems represented by nine scenarios combined among them [14].

The first step of the NeOn process is to define conceptual and functional requirements (detailed in Sect. 4.1). Conceptual requirements determine the concepts that should be present in the EdgeOnto and are used to analyze existing IoT, time-based, location-based, context-awareness and security ontologies detailed in Sect. 3.

Table 1 Existing ontologies review in the light of the identified requirements

Type	Ontology	CR1	CR2	CR3		CR4	CR5	CR6
				CR3.1	CR3.2			
Sensor	SSN	*	*	*	*	*	*	*
	SOSA	*	*	*	*	*	*	*
	SAREF	*	*	*	*	*	*	*
	SAN	*	*	*	*	*	*	*
	Onto Sensor	*	*	*	*	*	*	*
Time	DAML-Time	*	*	*	*	***	*	*
	DAML-S	*	*	*	*	***	*	*
	KSL-Time	*	*	*	*	***	*	*
	OWL-Time	*	*	*	*	***	*	*
	Timeline ontology	*	*	*	*	***	*	
Location	WGS84 ontology	*	*	**	*	**	*	*
Context	CONON	*	*	**	*	**	*	*
	Ontology of [3]	*	*	*	*	**	*	*
Security	STAC	*	*	*	*	*	***	*
QoS	Ontology of [31]	*	*	*	*	*	***	***
	QoS ontology [21]	*	*	*	*	*	***	***
IoT	IoT ontology	*	*	*	*	*	*	*
	IoT-Lite	*	*	**	*	*	*	*
	IoT-O	*	*	*	**	**	*	*

As recommended by NeOn, reusable ontologies that are compliant with parts of the requirements are integrated in our design process and presented in Sect. 4.2.

4.1 Functional requirements

Functional requirements regard the ontology structure and design principles. Reusability is an important aspect of ontology. Many approaches can be used to solve ontology reusability.

- Modularization: designing ontologies in separated modules makes them easier to reuse and/or extend [2]. In IoT applications, many domains can be integrated and it is difficult to capture them in the same ontology. According to specific needs and goals, modular ontologies can be combined together [29].
- Reuse of Existing Sources: in order to avoid redefinition and prevent redefined concepts from having to align a posteriori [29].
- Alignment to Upper Ontologies: The concepts expressed by upper ontologies are intended to be basic and universal to ensure generality and expressivity for a wide range of domains especially for IoT domain. These concepts are meta, generic, abstract and philosophical. The advantage of top-level ontologies is to gather lots of available knowledge and create super structures for information that provide interoperability for many applications [9].

4.2 Reused ontologies for EdgeOnto

For the design process of EdgeOnto, we reuse some existing ontologies: The IoT-O ontology [29], the description ontology for knowledge representation in the IoT domain [31], as well as, the DUL ontology.[2,3]

4.2.1 DUL ontology

DUL relies on DOLCE+DnS Ultralite ontology.[4] It is a simplification and an improvement of some parts of DOLCE Lite-Plus library,[5] and Descriptions and Situations ontology.[6] The DUL ontology main concepts are presented in Fig. 2.

The main abstract classes of DUL ontology are:

- DUL:Abstract: Any entity that cannot be located in space-time (like mathematical entities: formal semantics elements, regions within dimensional spaces, etc.).

[2] http://www.ontologydesignpatterns.org/ont/dul/DUL.owl.

[3] https://www.w3.org/2005/Incubator/ssn/wiki/Alignement_to_DUL_Upper_Ontology.

[4] https://ontologydesignpatterns.org/wiki/Ontology:DOLCE+DnS_Ultralite.

[5] https://dolce.semanticweb.org.

[6] https://www.ontologydesignpatterns.org/wiki/Ontology:DnS.

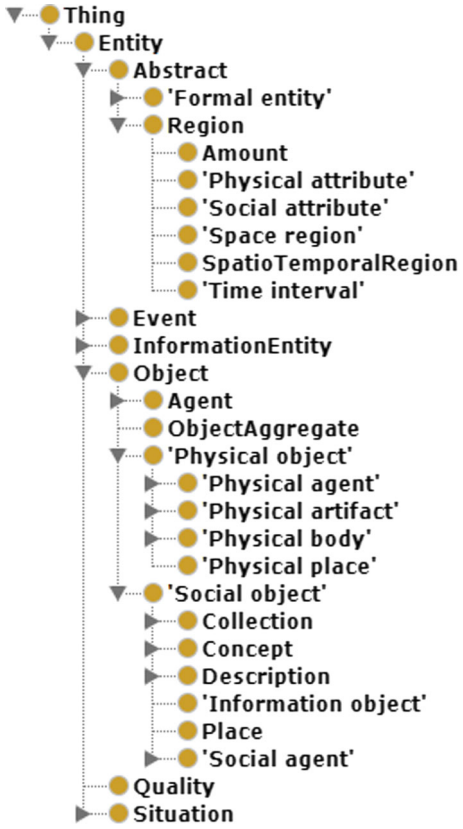

Fig. 2 DUL ontology tree concepts

- DUL:Event: Any physical, social, or mental process, event, or state. More theoretically, events can be classified in different ways, possibly based on 'aspect' (e.g., stative, continuous, accomplishment, achievement, etc.), on 'agentivity' (e.g., intentional, natural, etc.), or on 'typical participants' (e.g., human, physical, abstract, food, etc.).
- DUL:InformationEntity: A piece of information, be it concretely realized or not. It is a catchall class, intended to bypass the ambiguities of many data or text that could denote either an expression or a concrete realization of that expression.
- DUL:Object: Any physical, social, or mental object, or a substance. Following DOLCE Full, objects are always participating in some event (at least their own life), and are spatially located.
- DUL:Quality: Any aspect of an Entity (but not a part of it), which cannot exist without that Entity. For example, the way the surface of a specific PhysicalObject looks like, or the specific light of a place at a certain time, are examples of Quality, while the encoding of a Quality into, e.g., a PhysicalAttribute should be modeled as a Region.
- DUL:Situation: A view, consistent with ('satisfying') a Description, on a set of entities. It can also be seen as a 'relational context' created by an observer on the basis of a 'frame' (i.e., a Description).

For the rest of the paper, we have identified some conventions to follow. The concepts' names related to each ontology used are in bold and have the sans serif font different from the normal text. The concepts' names are proceeded by the name of the corresponding ontology.

4.2.2 IoT-O ontology

Reference [29] presents a core-domain modular IoT ontology with a vocabulary to describe connected devices and their relation with their environment. IoT-O is reused in our EdgeOnto because it is based on DUL ontology as well as other reference existing related to IoT (like SSN, life cycle, msm, PowerOnt, SAN). More informations are given about IoT-O in Sect. 2.6. The IoT-O OWL file is available at the following address.[7]

4.2.3 Knowledge representation ontology

Reference [31] presents an ontology for knowledge representation in the IoT domain and discusses how it can be used to support tasks such as service discovery, testing and dynamic composition (see Sect. 2.5). The main concepts reused of the description ontology [31] are QualityOfService, IoTService and IoTResource. For facilitating concepts' referencing, we give the acronym DO to the Description Ontology of [31]. The next section details the core domain EdgeOnto ontology applied for IoT domain in the context of edge computing.

5 EdgeOnto architecture and specifications

This section firstly discusses the several modules (i.e., IoT, location and time-awareness, and QoS management) that make up the EdgeOnto core domain. Our ontology is OWL-based (ontology web language). For each module, we detail the multiple defined concepts, as well as, the relations between them. Then, the second subsection formalizes the end user requests to query EdgeOnto. This is followed by the description of the associated semantic matching procedure.

5.1 EdgeOnto core-domain

EdgeOnto answers to the following questions: What domain EdgeOnto will cover (IoT domain applied to Cloud and Edge Computing)? For what EdgeOnto will be used (searching the nearest edge to the IoT device and choosing the most suitable

[7] https://www.irit.fr/recherches/MELODI/ontologies/IoT-O.html.

one)? For what types of queries, EdgeOnto should provide answers (proximity, similarity)? A deep analysis is undertaken to extract the main concepts related both to cloud, edge computing and its use in IoT. Related fields used in this analysis are fog computing, multi-access edge computing, time-based, location and IoT ontologies. The concept extracted is represented already by the conceptual requirements.

Secondly, an abstraction exercise is undertaken for identifying the main common concepts shared by various IoT applications and using cloud and edge computing. Concepts are organized as a class hierarchy where abstract concepts will be refined with more concrete ones specific to each domain application. They are also described with properties and connected to other concepts with semantic relations. We tried not to reinvent the wheel, so we further reuse existing ontologies (see Sect. 4.2).

The design process of EdgeOnto is based on:

- the formal specification of the functionalities achieved by EdgeOnto. It reuses both DUL, IoT-O and DO [31] ontologies and describe the main concepts related to IoT domain applied in the context of edge computing (see Sect. 4.1).
- the formal specification of what precisely the EdgeOnto needs/requires for identifying time, location and QoS, edge, cloud. In fact, EdgeOnto covers the conceptual requirements CR1, CR2, CR3, CR4, CR5 and CR6.

. After importing reused ontologies, a matching process is undertaken to integrate (i) new concepts to added and (ii) align concepts reused.

EdgeOnto is based on three modules:

- IoT module: describes the main concepts related to IoT (IoT-Thing, device, actuator, sensing device, etc.) and edge computing (edge, cloud).
- Time and location module: describes the main concepts related to time and location of each object in EdgeOnto (for example, location of an IoT-Thing in point of time, location of an edge node in instant t, location of a cloud node, etc.)
- QoS module: describes the quality of service between EdgeOnto objects (IoT devices, edge and clouds).

A set of semantic relations are defined in EdgeOnto represented in Table 2.

5.1.1 IoT module

This dimension encompasses four main concepts, namely EdgeOnto:Cloud, EdgeOnto:Edge, EdgeOnto:Resource and IoT-O:IoT-Thing (Fig. 3).

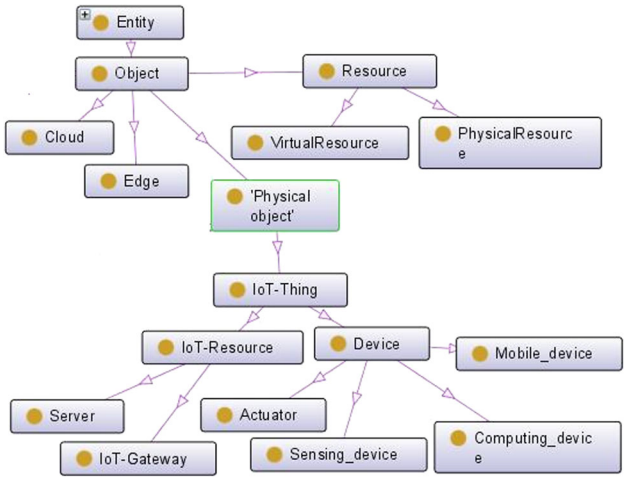

Fig. 3 EdgeOnto concepts related to IoT

- IoT-Thing refers to device (IoT-O:Sensing_device, EdgeOnto:Actuator, EdgeOnto:Mobile_device and EdgeOnto:Computing_device)
- DO:IoT-Resource which can be a server or an Iot - Gateway.
- A resource can be physical (EdgeOnto:PhysicalResource) or virtual(EdgeOnto: VirtualResource) and is used by a cloud or an edge.
- Cloud or edge (represented, respectively, by EdgeOnto by EdgeOnto:Cloud and EdgeOnto:Edge) are nodes programmed enabling recognition, processing or forwarding transmission to other nodes.
  - A node cloud (datacenter, database, server, etc.) is related to cloud computing which is defined by NIST as "*a model for enabling convenient, on demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction*".
  - An edge node (switcher, router, small/macro base station, etc.) is a node that acts as an end user portal for communication with other nodes (cloud node or device node).
  - Edge nodes are located between device (EdgeOnto:Device) and cloud (EdgeOnto:Cloud) and can facilitate computations nearer to the source of data (or where data is generated) and can incorporate strategies for remotely enhancing capabilities of front-end devices [30].

Table 2 EdgeOnto's semantic relations

Dimension	Relation	Range	Target
IoT	EdgeOnto:communicate	Edge	Cloud
	EdgeOnto:access	Device	Edge
	Dul:is used by	Resource	Object
Location, Time	DUL: is location of	Entity	Entity
	DUL: near to	Entity	Entity
	DUL: is region for	Region	Entity
QoS	DUL: offers	Object	Service
	EdgeOnto: hasQoS	Service	QualityOfService

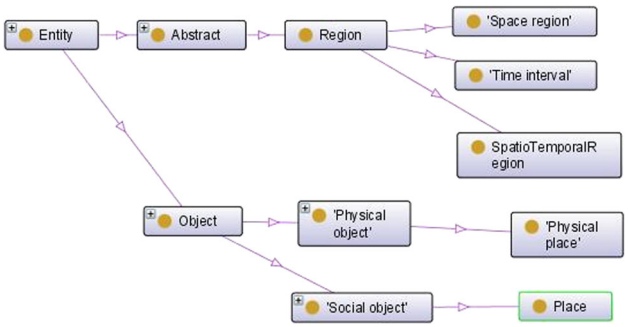

Fig. 4 Concepts related to location and time dimensions

5.1.2 Location and time-awareness module

The concepts proposed in this dimension are inherited from the DUL ontology (Fig. 4). For time dimension, we reuse mainly the concept of DUL:Time Interval defined as "any region in a dimensional space that aims at representing time".

For representing location, three types of place can be distinguished:

- Dependent location (represented by DUL: Place): is geographic entities and non-material locations determined by the presence of other entities or of pivot of events and signs as well as identified as complement to other entities. For example, the area where the mobile phone is located, surrounding of a temperature sensor
- Non-dependent location (represented by DUL: Physical place): can exist independently. It refers to a physical place where a physical object is inherently located. For example, a room, a building.
- Abstract or dimensional location (represented by DUL: Space Region): is any specific region in a dimensional space that is used to localize an entity (for example, cloud, edge node, device, etc.).

The semantic relation DUL: is region for is defined between DUL:Region and DUL: Entity. In EdgeOnto, this relation is available between DUL:Space Region, which is a DUL:Region and DUL: Entity.The inverse relation of DUL:Is region for is DUL:Has region

The semantic relation DUL:Has location is defined between DUL:Entity and DUL:Entity. In EdgeOnto, this relation is available between any entity (like EdgeOnto:Cloud, EdgeOnto:Edge, EdgeOnto:Device, etc.) and DUL: Place or DU:LPhysical place which are DUL:Entity. The inverse relation of DUL:Has location is DUL:Is location of. To ensure a consistent instantiation of concepts, Semantic Web Rule Language (SWRL) rules (including axioms) help enforce restrictions on attribute values and semantic relations, as well. Hereafter, we only exemplify SWRL rules referring to concepts. For example, Eq. 1 formally reflects the following statement: "Any EdgeOnto (?x) has location a place (?y) or a physical place (?z) and has region some space region (?p) during a time interval (?t)". Informally, each concept related to EdgeOnto (like Edge, Cloud or device) should have at least a location (which can be a physical place or a dependent place) during a time interval or a point of time.

$$\begin{aligned} &EdgeOnto(?x) \rightarrow \\ &hasLocation(?x, (place(?y, ?x) \mid \\ &physicalplace(?z, ?x))) \wedge \\ &hasRegion(?t, SpaceRegion(?x, ?p), \\ &TimeInterval(?t)) \end{aligned} \tag{1}$$

5.1.3 QOS management module

QoS is the description or measurement of the overall performance of a particularly the performance seen by the users of the network.

Different measurements are taken into account to characterize and evaluate QoS like availability, performance, reliability, scalability, and security.

The ontology proposed by [31] uses the DUL ontology and define the concept of Quality of Service (DO:QualityOfService) as a DUL:Information Object.

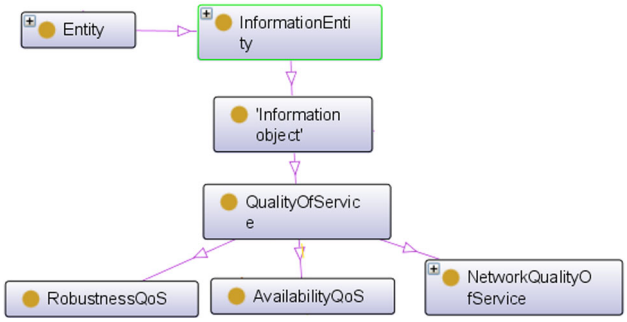

Fig. 5 Concepts related to QoS dimension

In IoT-O, reference [29] reuses also the DUL:Information Object which "is a piece of information, be concretely realized or not. It is intended to bypass the ambiguities of many data or text".

By aligning ontology of [31] and IoT-O, DO: QualityOfService is treated in this paper as an DUL:Information Object which is DUL:Information Entity (Fig. 5).

In IoT-O, a semantic relation IoT-O:hasQoS is defined between IoT-O:Service and DO:QualityOfService. In [29], IoT-O:Service is "a set of operation and provides a user a way to issue requests through an interface. Underlying implementation need not to be known by the end user".

SWRL rules are defined to infer new semantic relations between instances during concept instantiation related to QoS dimension. For instance, Eq. 2 formally reflects the following statement: "Any EdgeOnto (?x) that offers a service (?y) should has a specific quality of service (?u)". Informally, each EdgeOnto concept defined as an object (like device, cloud, or edge) offers a certain service which has a quality. The quality of service offered is already measured by different metrics in the literature.

$$EdgeOnto(?x) \wedge offers(?x, Service(?y)) \rightarrow hasQoS(?x, QualityOfService(?z, ?u)) \quad (2)$$

- x corresponds to the EdgeOnto instance(s) to be retrieved.

5.2 User request building

We define requirement (REQ) as a set of concepts in EdgeOnto requested by the user U_i. We have already detailed the set of requirement in Sect. 3. Formally, Eq. 3 represents the syntax used for specifying REQ.

$$REQ_i = EdgeOnto(?x)[\wedge Concept_j(?x, y)]_{j=1..n} \quad (3)$$

where

- $Concept_j \in$ EdgeOnto (concepts related to this ontology).
- Note U_i can refine EdgeOnto into concrete concepts related to a specific domain.

5.3 Semantic matching

Semantic matching is a technique used to identify information (concepts in the case of ontology) which is semantically related. The semantic match maker algorithm takes an OWL-S request for the user as input and iterates every OWL-S advertisement in its repository in order to determine a match [6]. In the OWL-S approach, functionality of a service is described in terms of inputs, outputs, preconditions and effects. Input and output terms of the service are expressed as concepts belonging to a set of ontologies. An advertisement (Advt) and a query (Query) match if their outputs and inputs match.

- For every input parameter in Advt, there is one input parameter in query. Let $Query_{in}$ and $Advt_{in}$ represent the list of input concepts of query and the advertisement, respectively. The service can correctly perform the task if all the input concepts defined in the advertisement are satisfied by the requester (Eq. 4).
- For every output parameter in Query, there is one output parameter in Advt. Let $Query_{out}$ and $Advt_{out}$ represent the list of output concepts of query and the advertisement, respectively. The service can be used by the requester if all the output concepts defined in the query are satisfied by the advertisement (Eq. 5).

$$\forall c \in Advt_{in}, \exists d \in Query_{in}, s.t.match(c, d) \neq Fail \quad (4)$$

$$\forall c \in Query_{out}, \exists d \in Advt_{out}, s.t.match(c, d) \neq Fail \quad (5)$$

6 Use case implementation and experimentation

This section briefly describes the considered test collection and presents the EdgeOnto's population. Then, it discusses the performed experiments to evaluate and validate our findings. Finally, it presents the obtained measurements in terms of performance and robustness.

6.1 Test collection

Table 3 shows an excerpt of EdgeOnto's population. There are three dimensions: IoT, location, time and QoS dimensions. To conduct experiments on EdgeOnto semantic discovery, we first proceed with the test collection creation.

Table 3 Excerpt of EdgeOnto's population

Dimension	Concept	Instances
IoT	Edge	Collector node, Collector node1
	Server	VNC server, VNC viewer
	Physical Resource	Raspberry Pi 4B, InfluxDB, Grafana, Creality Ender Pro
	Virtual Resource	GRAD-CAM algorithm
	Sensing-device	Humidity and temperature sensor DHT11, Soil humidity sensor hygrometer, HD USB camera
	Computing-device	YOLO 5
	Mobile device	Smart agriculture drone 1, Smart agriculture drone 2, robot 1
Location, Time	Physical place	Strawberry farm1,place1, place2
	Space region	Position (50,40), position (100.120)
	Place	Middle of the strawberry farm,at the end of strawberry farm
	TimeInterval	Duration between the informations' reception coming from the node collector and its treatement on the node collector, 13am
QoS	QualityOfService	Quality of communication between collector node and device node, quality of identifying humidity and temperature measurements, quality of computing soil humidity
	Service	Computing humidity and temperature, computing soil humidity

Table 4 Excerpt of EdgeOnto's user request

Query	User request
REQ1	$\forall Device(?x) \wedge nearTo(?x, Edge(?y)$
REQ2	$\forall Device(?x) \wedge access(x?, (collectorNode \vee collectorNode1))$
REQ3	$\forall Device(?x) \wedge access(x?, (collectorNode \vee collectorNode1)) \wedge hasLocation(?x, Place(?z) or PhysicalPlace(?p))$
REQ4	$\forall Device(?x) \wedge access(x?, Edge(?y)) \wedge offers(x?, Service(?z))$
REQ5	$\forall Device(?x) \wedge access(x?, Edge(?y)) \wedge offers(x?, Service(?z)) \wedge hasQoS(?z, QualityOfService(?q))$
REQ6	$Service(?x)) \wedge hasQoS(?x, QualityOfService(?y))$
REQ7	$Place \wedge isLocationOf(?x, Entity(?y))$
REQ8	$Place(?x) or Physicalplace(?y) or Spaceregion(?z) \wedge isLocationOf(?x, Entity(?y))$
REQ9	$Resource(?x) \wedge isUsedBy(?x, Object(?y))$
REQ10	$Entity(?x) \wedge' hasregion'(?x,' Spaceregion'(?y)$

For a clarity purpose, Table 4 presents an excerpt of the user request identified in the smart strawberry farming use case introduced in the Sect. 1.

SWRL rules are defined to infer new semantic relations between instances during EdgeOnto's population.

For instance, Eq. 6 states that "Any Device (?x) that offers a Service (?y) at a point of time or Interval Time (?t) located on a Physical place or Place or Space region (?l) should access to some Edge (?z) the nearest to this device.

$$\begin{aligned} &\forall Device(?x) \wedge offers(?x, Service(?y)) \\ &\quad \rightarrow \exists Edge(?z) \wedge access(?x, Edge(?z)) \\ &\qquad\qquad \wedge nearTo(?x, Edge(?z)) \end{aligned} \tag{6}$$

6.2 Performance analysis

To assess the proposed approach's performance, we use two metrics, namely completeness and efficiency. The former describes how well our Protégé matchmaker identifies the relevant EdgeOnto concepts compared with the total number of such EdgeOnto concepts that exist in the test collection. The latter describes how well Protégé identifies only those relevant EdgeOnto concepts, by comparing the number of target EdgeOnto-identified concepts with the total number of EdgeOnto-retrieved concepts. The main preference metrics are True Positive (TP), False Positive (FP), and False Negative (FN) where

- TP contains the retrieved EdgeOnto concepts that are relevant
- FP contains the retrieved EdgeOnto concepts that are not relevant
- FN contains the relevant EdgeOnto concepts that are not retrieved (i.e., discarded by the matchmaker).

Once the sets mentioned above are established, two popular performance measurements, in the semantic web and machine learning communities, are calculated, namely recall and precision. These performance measurements implement completeness and efficiency metrics and are defined, respectively, as follows:

- Recall is the quantity's measure in the response (how close was the result to the actual response). It refers to the ratio between the number of true-positive EdgeOnto concepts and the number of relevant EdgeOnto concepts, including true-positive EdgeOnto concepts and false-negative EdgeOnto concepts (Eq. 7).

$$Recall = \frac{TP}{TP + FN} \tag{7}$$

- Precision is the quality's measure of the response (how much the response is correct). It refers to the ratio between the number of true-positive EdgeOnto concepts and the total number of retrieved EdgeOnto concepts, including true-positive and false-positive EdgeOnto concepts (Eq. 8).

$$Precision = \frac{TP}{TP + FP} \tag{8}$$

6.3 Robustness evaluation

This section discusses the performed experiments to evaluate and validate our findings. It presents the analyzed obtained measurements in terms of performance and robustness by applying Eqs. 8 and 7:

$$\text{Precision} = 0.875 \text{ and Recall} = 0.954$$

These measurements depict that EdgeOnto reach good accuracy for requests' response (75%) Compared with other IoT ontologies, none of the user requests mentioned in Table 4 can give relevant responses and satisfy user requests. This is justified by many reasons:

- Lack of concepts in relation with mobility
- Lack of concepts in relation with context, location and time awareness
- Lack of concepts related to edge, cloud, and device types.
- Lack of concepts describing quality of service .

In fact, the proposed ontology EdgeOnto clearly outperforms the other IoT ontologies and can provide much better accuracy.

7 Conclusion and future work

IoT domain relies, nowadays on hybrid cloud/edge environment for faster communication, lower bandwidth and better local treatment.

Compared with existing works, most of them don't satisfy the requirements already defined about time, location, mobility, and quality of service.

In this paper, we propose a semantic model (EdgeOnto) which highlights all concepts related to IoT applied in the context of edge computing. Its main goal is to support the automation of the QoS management procedures in hybrid cloud/edge environment and the discovery of the relevant edge nodes that are suitable to host and execute IoT services considering their requirements.

The illustrative use case is smart strawberry farming [12], and the proposed edge platform aims to be an all-in-one IoT platform to enable the intelligent farm on strawberries cultivation with wireless sensor network (WSN), computer vision (CV), machine learning (ML), and long-range (LoRa) communication capabilities. The platform makes available to the user all the captured metrics for manual analysis and data-driven decisions. This platform studies only the communication between devices and edges where devices and edges are static and have the same location during the strawberry farming scenario. In addition to these limitations, the current edge framework lacks of automation when implementing steps that make up the IoT services life cycle in hybrid cloud/edge environment. Edge nodes are often manually selected during deployment time and most of the regular QoS management procedures remain difficult to implement.

In future works, we study deeply the problem of optimal placement where edge and device are simultaneously mobile. In fact, when device node is mobile, we should find the optimal edge node the nearest to this device node ensuring a sufficient and acceptable quality of service.

Declarations

Conflict of interest No conflicts of interest relevant to content presented in this article are associated. The authors have no relevant financial or non-financial interests to disclose, nor competing interests to declare that are relevant to the content of this article or could have influenced its outcome.

References

1. Ankolekar A, Burstein M, Hobbs J, Lassila O, Martin D, Mcdermott D, Mcilraith S, Narayanan S, Paolucci M, Payne T, Sycara K (2002) 06. Daml-s: Web service description for the semantic web. Science 6:97
2. Aquin M (2012) Modularizing ontologies. In: Suárez-Figueroa MC, Gómez-Pérez A, Motta E, Gangemi A (eds) Ontology engineering in a networked world. Springer, Berlin, pp 213–233
3. Bae IH (2014) 04. An ontology-based approach to adl recognition in smart homes. Futur Gener Comput Syst 33:32–41
4. Bajaj G, Agarwal R, Singh P, Georgantas N, Issarny V (2017) 07. A study of existing ontologies in the iot-domain. arXiv:1707.00112
5. Baldauf M, Dustdar S, Rosenberg F (2007) A survey on context-aware systems. Inf Syst 2(4):263–277
6. Bellur U, Vadodaria H, Gupta A (2008) 11. Semantic matchmaking algorithms. INTECH Open Access Publisher, London, pp 481–502
7. Bermúdez-Edo M, Elsaleh T, Barnaghi P, Taylor K (2015) 11. Iot-lite ontology. W3C Memb Submiss 5:26
8. Bibani O, Yangui S, Glitho RH, Gaaloul W, Hadj-Alouane NB, Morrow MJ, Polakos PA (2016) A demo of a paas for iot applications provisioning in hybrid cloud/fog environment. In: IEEE international symposium on local and metropolitan area networks, LANMAN 2016, Rome, Italy, June 13–15, 2016, pp 1–2. IEEE
9. Bontas EP, Mochól M, Tolksdorf R (2005) Case studies on ontology reuse. In: IKNOW05 international conference on knowledge management, vol 74, pp 345
10. Canete A, Amor M, Fuentes L (2022) Supporting IoT applications deployment on edge-based infrastructures using multi-layer feature models. J Syst Softw 183:111086
11. Compton M, Barnaghi P, Bermudez L, García-Castro R, Corcho O, Cox S, Graybeal J, Hauswirth M, Henson C, Herzog A, Huang V, Janowicz K, Kelsey WD, Le Phuoc D, Lefort L, Leggieri M, Neuhaus H, Nikolov A, Page K, Passant A, Sheth A, Taylor K (2012) The SSN ontology of the W3C semantic sensor network incubator group. J Web Seman 17:25–32
12. Cruz M, Mafra S, Teixeira E, Figueiredo F (2022) Smart strawberry farming using edge computing and IoT. Sensors 22(15):740
13. Daniele L, Solanki M, den Hartog F, Roes J (2016) 10. Interoperability for smart appliances in the iot world, pp 21–29
14. Del Carmen Suárez de Figueroa Baonza, M (2010) NeOn methodology for building ontology networks: specification, scheduling and reuse. Ph.D. thesis, Universidad Politécnica de Madrid, Madrid, Spain
15. Flury T, Privat G, Ramparany F (2004) Owl-based location ontology for context-aware services. Proc Artif Intell Mob Syst 7:52–57
16. Gyrard A, Bonnet C, Boudaoud K (2013) 05. The stac (security toolbox: attacks and countermeasures). Ontology 5:165–166
17. Hobbs J, Pan F (2004) 03. An ontology of time for the semantic web. ACM Trans Asian Lang Inf Process 3:66–85
18. Hobbs JR (2002) A daml ontology of time
19. Janowicz K, Haller A, Cox S, Phuoc D, Lefrançois M (2018) 07. Sosa: a lightweight ontology for sensors, observations, samples, and actuators. J Web Seman 56:1–10. https://doi.org/10.1016/j.websem.2018.06.003
20. Jerry R, Hobbs FP (2006) Time ontology in owl
21. Jiang S, Aagesen F (2006) 01. An approach to integrated semantic service discovery, vol 4195, pp 159–171
22. Li X, Zhou Z, Zhao Z, Yangui S, Zhang W (2021) Data & computation-intensive service re-scheduling in edge networks. In: Chang CK, Daminai E, Fan J, Ghodous P, Maximilien M, Wang Z, Ward R, Zhang J (eds) 2021 IEEE international conference on web services, ICWS 2021, Chicago, IL, USA, September 5–10, 2021. IEEE, pp 389–396
23. Martinez-Villase nor L, Gonzalez-Mendoza M (2014) 11. Sharing and reusing context information in ubiquitous computing environments. pp 227–230
24. Mouradian C, Naboulsi D, Yangui S, Glitho RH, Morrow MJ, Polakos PA (2018) A comprehensive survey on fog computing: state-of-the-art and research challenges. IEEE Commun Surv Tutor 20(1):416–464
25. Nachabe L, Girod-Genet M, ElHassan B (2015) 01. Unified data model for wireless sensor network myontosens ontology. IEEE Sens J 7:3657–3667
26. Nicolas S, Mahdi BA, KDNHTM (2015) San
27. Ntalasha D, Renfa L, Wang Y (2016) 02. Internet of thing context awareness model. EAI Endors Trans Context Aware Syst Appl 3(7):151084
28. Russomanno D, Kothari C, Thomas O (2005) 04. Sensor ontologies, from shallow to deep models, pp 107–112
29. Seydoux N, Drira K, Hernandez N, Monteil T (2016) 11. Iot-o, a core-domain IoT ontology to represent connected devices networks, pp 561–576
30. Varghese B, Wang N, Barbhuiya S, Kilpatrick P, Nikolopoulos DS (2016) Challenges and opportunities in edge computing. In: 2016 IEEE international conference on smart cloud (SmartCloud), pp 20–26
31. Wang W, De S, Toenjes R, Reetz E, Moessner K (2012) A comprehensive ontology for knowledge representation in the internet of things. In: 2012 IEEE 11th international conference on trust, security and privacy in computing and communications, pp 1793–1798
32. Wang X, Zhang D, Gu T, Pung H (2004) 01. Ontology based context modeling and reasoning using owl, pp 18–22
33. Xue L, Liu Y, Zeng P, Yu H, Shi Z (2015) 08. An ontology based scheme for sensor description in context awareness system. In: 2015 IEEE international conference on information and automation, pp 817–820
34. Yangui S (2020) A panorama of cloud platforms for IoT applications across industries. Sensors 20(9):2701