WoR Ontology: Modeling Resources in Web Connected Environments

Lara Kallab, Richard Chbeir, Sana Sellami

WoR Ontology: A Web Resource Model for Connected Web Environments

Lara Kallab[1], Richard Chbeir[2], and Sana Sellami[3]

[1] Open Group, Levallois Perret, 92300, France
lara.kallab@open-groupe.com
[2] Univ Pau & Pays Adour, E2S UPPA, LIUPPA, EA3000, Anglet, France
richard.chbeir@univ-pau.fr
[3] Aix Marseille Univ, Université de Toulon, CNRS, LIS, Marseille, France
sana.sellami@univ-amu.fr

Abstract. The Web of Things (WoT) describes a set of standards by the World Wide Web Consortium (W3C) for the interoperability of different Internet of things (IoT) platforms, and different application domains. Thus, it guarantees not only device-to-device interactions but also application-to-application communications, despite their platform heterogeneity. To identify and use the services (also called resources) that are exposed by either the devices or the applications connected to a Web environment, describing them using an open, shared and dynamic knowledge representation is required, allowing them to interoperate on both syntatic and semantic levels. In this paper, we propose WoR, a Web of Resources ontology that provides a modular and a common vocabulary to describe Web resources. WoR can: (1) ease the discovery, the selection, and the composition of the different kind of resources (exposed by connected Web devices or Web applications), (2) provide reasoning means to discover new information, and (3) allow future extensibility and adaptation to new domains needs. Experiments were made to evaluate our proposed WoR ontology, showing promising results on the accuracy, clarity, and performance levels.

Keywords: Web of Things · Web Resource · Semantic Resource Model. Web Resource Composition

1 Introduction

The Internet of Things (IoT) is a concept that encompasses a wide range of objects (e.g., smart devices), embedded with electronics, software, and sensors, that are connected to the internet to collect and share data [22]. With the heterogeneity of objects trying to communicate with one another through various ways of interaction, it is challenging to seamlessly build a single communication platform, in which all of the objects are interoperable and can communicate together effectively. The same challenge appears in application-to-application interactions, as applications may be developed on different platforms and may use various programming languages. As a way to tackle such challenge, the Web [17]

has emerged as a technology that guarantees the communication between the different connected objects and applications, despite their heterogeneity. Since then, the Web has become a major medium of communication platform [16], that leads to the emerging of the Web of Things [2] (WoT). In WoT, IoT objects are integrated into the Web, enabling them to expose their functions in the most efficient manner in spite of their diversity. The Web platform has been also broadly adopted to expose applications functions, to allow for language and platform independence and to improve interoperability with other applications.

In the Web context, the functions of the objects and applications are provided via Web services, i.e., self-contained, self-describing, modular components that can be published, located, and invoked across the Web [23]. The REST architectural style [21] has recently become the most used solution for implementing Web services [10]. This is due to: (i) its simplicity and ease of use that make services integration cost-effective, (ii) its support for different data formats (e.g., JSON[4] and XML[5]), and (iii) its ability to support caching for better performance and scalability. Hence, more and more objects and applications provide their functions as REST-based Web services that follow the principles of the REST architectural style. These Web services should be defined in a way that allows them to be largely adopted for interoperability purposes. The Web alone does not solve the interoperability issues. In fact, it is well acknowledged that semantic Web technologies hold the potential of achieving data and platform interoperability [13]. This is done by describing and sharing knowledge on Web services using unambiguous and machine-understandable vocabularies that attribute the same meaning to the specified and exchanged data. One of the building blocks in semantic Web Technologies, is the "Ontologies" [20], which provide a common and a comprehensible vocabulary for publishing data through a formal explicit specification of a domain conceptualisation. They facilitate knowledge sharing between systems across different organizations, allowing them to interoperate by solving the problem of interoperability on both syntatic and semantic levels.

Over the last decade, many semantic-based models (especially ontologies-based models) [24][27][14][12] have been built to specify IoT/WoT domains, including their exposed services (functions). These models are defined for different reasons, among them, to represent a specific domain knowledge (e.g., specifying the entities, relations, and activities involved in sensing data by smart devices) in a machine-understandable way, allowing things with their functions (or services) to be discoverable, aggregated, and remotely accessible. However, these models have several limitations while describing the provided services. As such, most models [9][12][27] have been defined to focus on describing the services of connected objects, without considering the ones that can be provided by applications. Also, and despite that the majority of the models (as in [5] and [7]) include important concepts to describe a service provided by an object (as object locations, their provided functions with their necessary inputs and outputs), they lack in considering other important aspects, such as services semantic links

[4] JavaScript Object Notation, http://json.org/
[5] Extensible Markup Language

(e.g., similarity links which denote that a service is similar to another one as they provide the same function). The services semantic links can be useful while composing services, where services are linked together to form new value-added services [26], when no single service can answer certain demands. Moreover, within this context, most of the models do not include the description of composed Web services, which can be used later in other different scenarios. And if in few models it is included, such description is limited, and does not follow the REST architecture style principles [11]. The latter criteria is important to consider as REST-based Web services are lighter for IoT/WoT devices comparing to SOAP-based Web services [25], which are considered by most of the current existing models. In addition, recently, the concept of virtual devices (e.g., virtual sensors [8]), has been emerged, to aggregate capabilities of IoT devices and derive new services. These new services, referred to as virtual services in this paper, and which are considered as value-added services besides the composed services as they allow to simulate the behavior of services of the devices that can not really exist, are not included within the existing IoT/WoT models.

To address the aforementioned limitations, we propose in this paper, an ontology-based Web resource model, called WoR (standing for Web of Resource), for connected Web environments, in which objects and applications can be connected to the Web, to provide and exchange data. WoR provides a common vocabulary that is used to describe exposed Web resources (REST-based Web services) by both objects and applications. It is able to: (i) store and integrate resource specifications related to heterogeneous objects and applications, (ii) ease the discovery, selection, and composition of the exposed resources, (iii) provide reasoning means to discover new information, and (vi) allow future extensibility and adaptation.

The rest of the paper is organized as follows. Section 2 presents a scenario to motivate the usability and applicability of our work. Section 3 presents the related work, and shows the originality of our model. Section 4 details the specifications of our proposed Web resource model. Section 5 evaluates and validates the solution. Finally, Section 6 summarizes the work and gives future directions.

2 Motivating Scenario

In order to motivate our work, we choose a scenario that can be applicable in OpenCEMS[6]: a Web platform that provides solutions for energy data management in connected environments. The platform allows connecting objects (stationary or mobile) and Web applications, both exposing Web resources for either: collecting on-site data, or preprocessing collected data, or analyzing data. In the scenario, we consider a facility manager of a smart mall connecting a set of smart sensors (e.g., temperature sensors and humidity sensors), and providing several Web applications (e.g., temperature and energy data predictions). The facility manager wants to predict the energy consumption of a specific floor of

[6] Connected Environment & Distributed Energy Data Management Solutions: `https://opencems.sigappfr.org/`

the mall, for the upcoming 2 days. The prediction results can help him monitor the energy consumption of the corresponding floor, and allow for establishing plans for energy supply and demand. Through a visualization interface provided

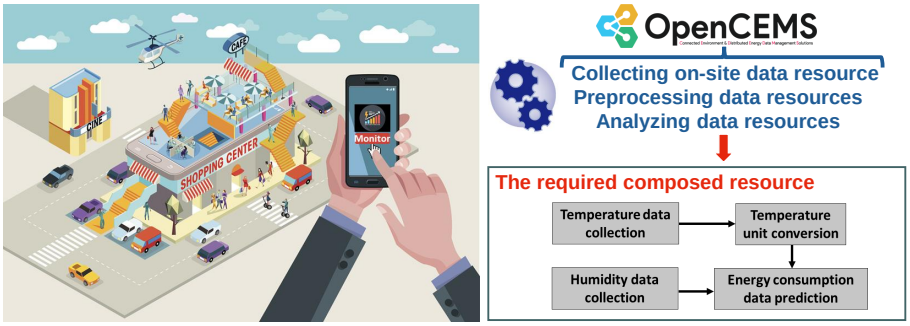


Fig. 1: An OpenCEMS instance applied in a smart shopping center, with an example of a composed resource

by OpenCEMS that is adapted to the smart shopping center, the facility manager can visualize the set of the resources provided by the mall, to call the most convenient ones answering his demand. In order to satisfy his need, several resources are to be selected and linked together to form the necessary composed resource, as shown in Figure 1: (i) data collection resources to collect the necessary data (e.g., external temperature), (ii) preprocessing data resources to prepare the collected data (e.g., temperature unit conversion), and (iii) advanced data processing resources, such as the resource used to predict the energy consumption data. However, the existing of numerous and heterogeneous resources providing different functions, but also, in some cases, having the same required functions, along with their different characteristics in terms of response time, performance, etc., make resource identification and selection complex tasks for the facility manager. To facilitate these tasks, it is important to describe the exposed resources through a unified description model, that is expressed in a human-comprehensible and a machine-readable vocabulary. Such model should cover the following criteria:

1. **A thorough model**. Due to the various types of resources that can be connected to the shopping mall, it is important that the model allows their description to meet the different facility manager needs.
   - **Object and application resources:** As the shopping center can connect objects (e.g., smart sensors) and Web applications, it is important that the model allows the description of the resources exposed by these two different sources. In fact, the resources can be described using common concepts, such as the “Function" concept, which is a necessary to consider for all kind of resources (i.e., whether exposed by a temperature sensor or by an energy consumption prediction application), to know

the exact function they provide. However, they can have, each, specific concepts, e.g., "Location" and "Operation Range", which can only be assigned for the installed sensors.

- **Composed resources:** In some cases, there is no single resource that can satisfy a specific user demand. However, the combination of two or more resources, forming a composed resource (a composition), may provide the required outputs. This is the case of the facility manager demand that requires the use of several resources (for data collection, data preprocessing, and advanced data processing). In order to enable the use of already formed composed resources and avoid repeating the composition process from the start, which consumes time and resources (CPU, memory, etc.), it becomes necessary to model and store the composed resources. Figure 2 shows an example of a resource composition formed by several resources, that can answer the facility manager demand. To avoid repeating the composition process, and allow the facility manager to use it later on, storing its necessary workflow, defined by the set of the used resources with their links together, is important. As shown in Figure 2, the stored composed resources can be considered as single resources to be used in other scenarios.

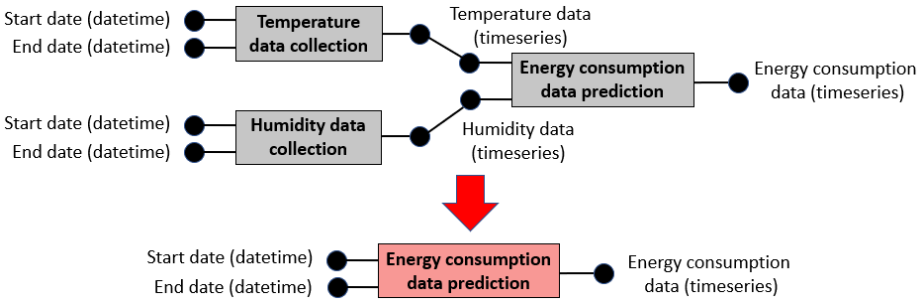


Fig. 2: A stored resource composition behaving as a single resource

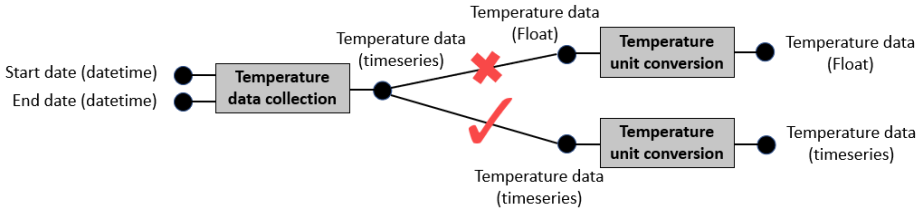
- **Virtual resources:** Some users demands may require to use data that cannot be collected by actual physical devices. For example, the facility manager may desire to collect temperature data from sensors that are not physically placed in the corresponding floor (e.g., due to high cost reasons). For such matters, virtual resources can be used to simulate the behavior of these physical sensors, by using some existing data and applying few calculations, in order to acquire the necessary results. Thus, allowing the description of virtual resources gets to be efficient for several cases.

2. **An expressive model**. Several description aspects should be taken into consideration in order to represent, at best, each provided resource, to allow

its correct usage in different scenarios. Below we motivate the use of these aspects.

- **Provided function:** It is the first main aspect to consider for each resource, to help in identifying the suitable resources that can answer the facility manager need. For example, when requesting to predict energy consumption data, it is important to be able to identify the set of resources realizing such function.
- **I/O:** Describing the Inputs/Outputs (I/O) of a resource allows to know the expected/returned data for/from each resource. As such, lets consider the phase when the facility manager wants to collect the necessary external temperature data surrounding the shopping center in °C, and which will be later used to predict the energy consumption data required for the corresponding floor. Figure 3(a) shows two possible resources that can be used to convert the collected external temperature data to the necessary unit, but each, requires different inputs. In order to select the right one realizing user need, describing the necessary inputs required for each resource is important. Moreover, resources I/O help in ensuring that the resources are efficiently linked together while composing a resource. Figure 3(b), for instance, illustrates an example of an I/O type mismatch between two linked resources that might affect the quality of the predicted results.

(a) Two resources providing the same function but requiring different inputs



(b) I/O type mismatch between linked resources

Fig. 3: Examples showing the importance of describing resources I/O

- **Location:** In connected environments, like the smart shopping center, several devices (stationary/mobile) are used to collect data. In order to collect relevant data and provide pertinent predicted results, it is crucial to consider devices location. As such, when predicting the energy consumption of a specific floor of the shopping center, it is important to collect the necessary data (humidity, temperature, etc.) from the set of devices that are located in that floor.

- **Links:** Defining links between the provided resources of an environment, can facilitate resource identification and selection, as well as ease resource replacement whenever a resource is no more available. For instance, when a resource is defined to be complementary to another (i.e,. provides a function that is dependent of the other), the facility manager can know what possible resources can choose after selecting a specific resource. This is the case of the resource “Energy consumption data prediction" shown in Figure 4, which is complementary to the “Temperature data collection" resource. And when a resource is defined to be similar to another (i.e., provides the same function), the facility manager can have several resource options (candidates) that satisfy his need. The similarity links between resources can also help in replacing a non-available resource (e.g., disconnected from the environment) by another similar one.

Fig. 4: Semantic links examples between resources

3. **A model supporting resource quality aspects**. When several connected resources provide the same needed function to answer the facility manager demand, it is quite challenging to distinguish the most convenient one to his request, among the others. Therefore, it is important to differentiate between candidate resources having similar functions, by defining non-functional aspects, referred to as Quality of Service (QoS), which is in our case Quality of Resource (QoR). However, with the presence of resources that can be provided by either objects or applications, several QoR levels are to be considered:
   - **Physical level:** refers to the attributes related to the physical devices (e.g., Operation Range, Battery, etc.). As such, the facility manager may prefer to collect temperature data using sensors that have the highest operation range to acquire the most pertinent collected ambient data.
   - **Network level:** designates the attributes related to the communication/transferred data between the devices/the application (e.g., Bandwidth, Latency, etc.). For example, the facility manager may require fast results, which can be affected by the amount of data transmitted between objects and applications. Thus, considering the Bandwidth or/and the Latency of the communicated data in the network is important.
   - **Application level:** represents the attributes related to the quality of services provided by each resource exposed either by a device or an application (e.g., Usage, Response Time, etc.). In this context, the facility

manager may require using the resources that have been used many times, denoting a high usage rate. As such, the more a resource is called to answer user demands, the more it proves its efficiency in several scenarios.

In this paper, we embed the aforementioned criteria in a semantic Web resource model (ontology) explained in details in Section 4. The proposed ontology model is expressed using a vocabulary that is human-comprehensible to facilitate the identification, the selection, and composition of resources by end-users. Such vocabulary is also machine-readable that allows to automatically identify, select, and compose resources.

3 State of the Art

In this section, we study several IoT and WoT-based services (or resources) models, and compare them according to different criteria. The criteria, grouped into 3 categories (as shown below), are mainly related to the concepts/properties used to model the services exposed by the connected "Things" (e.g., Web objects and Web applications):

1. **Thorough model:** denoting the ability of the model to describe different types of services:
   - Objects and applications services, referring to the services that can be either exposed by connected objects, or connected applications
   - Elementary services, referring to the services that are not linked to any other resource, and whose behavior is not simulated
   - Complex types of services, i.e., composed services and virtual services
   - Categories, referring to the services categories (e.g., data collection services and data preprocessing services), which can facilitate the exploration of services, and the understanding of their behavior
2. **Expressiveness:** indicating the ability of the model to cover various criteria representing the resource:
   - Provided function
   - I/O, referring to the inputs and outputs of the services
   - Location, referring to the object location (whenever the services are exposed by connected objects)
   - Links, designating the links between the services provided by the connected objects/applications
3. **Resource quality:** designating the ability of the model to define resource qualities at various levels:
   - Physical, referring to the physical properties of the connected objects exposing services
   - Network, denoting the quality aspects of the data transfer/communication that can be supported by the services
   - Application, referring to the quality of the service provided by each resource

3.1 IoT/WoT-based Models

IoT-O [24], is a core-domain modular IoT ontology that proposes a vocabulary to describe connected devices and their relation with their environment. As shown in Figure 5, it includes five modules: (1) the Sensing module, which is based on SSN ontology [6] to describe the sensors and their observations, (2) the Acting module, which is based on SAN ontology[7] to describe how IoT devices can interact with the physical world (i.e., their performed actions), (3) the Lifecycle module that models state machines to specify IoT devices life cycle and usage, (4) the Service module, which describes the services provided by the IoT devices using MSM[8], a REST architecture style [11]-based ontology, and (5) the Energy module that is defined by PowerOnt [4] to express power consumption profiles for appliances.

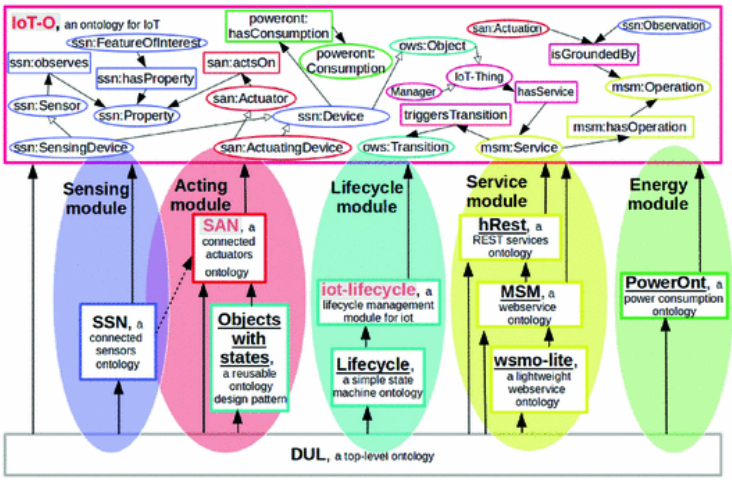


Fig. 5: IoT-O main modules and concepts

DUL[9] is an upper ontology that is used by these 5 modules. In the IoT-O model, any connected element can provide a Service, whether it is a physical object (ssn: Device) or an application (iot-o: Manager). Each Service is identifiable by an address, and can provide some Operation callable using HTTP method (e.g. GET and PUT). A method may have a set of inputs, outputs, and hypertexts to link the outputs of the operation to other operations.

oneM2M[10] is a global standard for Machine to Machine Communications and the IoT, developed in an open and collaborative manner by many companies.

[7] https://www.irit.fr/recherches/MELODI/ontologies/SAN.html
[8] https://lov.linkeddata.es/dataset/lov/vocabs/msm
[9] http://www.ontologydesignpatterns.org/ont/dul/DUL.owl
[10] https://www.onem2m.org/technical/onem2m-ontologies

oneM2M allows to annotate application specific resources (M2M data) with semantic description. It specifies a top-level Base ontology, illustrated in Figure 6, that allows to create sub-classes (or equivalence classes) for application-level ontologies, e.g., Smart Appliances REFerence Ontology (SAREF)[11]. In the Base ontology of oneM2M, a Device has a Service that exposes some Operation. An Operation has some Operation Input and Operation Output, which are Variables that describe an Aspect, e.g., Temperature.

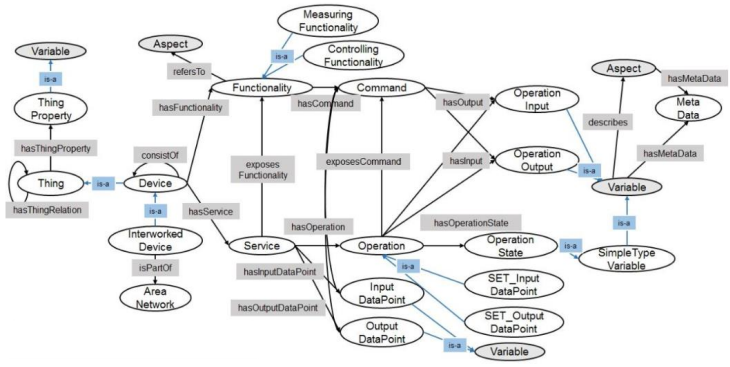


Fig. 6: oneM2M Base ontology

SAREF ontology[12] is a modular and domain-independent semantic layer for smart appliances. It explicitly specifies recurring core concepts in the smart applications domain, the main relationships between these concepts, and axioms to constrain the usage of these concepts and relationships. Figure 7 shows an overview of the SAREF ontology. In SAREF, a Service can represent one or more Function offered by a Device. A Service also specifies the Device that is offering the Service, the Function to be represented, and the input and output parameters necessary to operate the Service.

WoT TD (Thing Description) [5] is a formal model and a common representation for the Web of Things that is defined within W3C's WoT working group. A Thing Description describes the metadata and interfaces of Things, where a Thing is an abstraction of a physical or virtual entity that interacts and participates in the Web of Things. Thing Descriptions provide a set of interactions based on a small vocabulary that makes it possible to integrate diverse devices and to allow diverse applications to interoperate. Thing Descriptions can be encoded either in a JSON format or in a JSON-LD description to represent the knowledge about things in a machine-understandable way.

[11] https://saref.etsi.org/core/v3.1.1/
[12] https://saref.etsi.org/

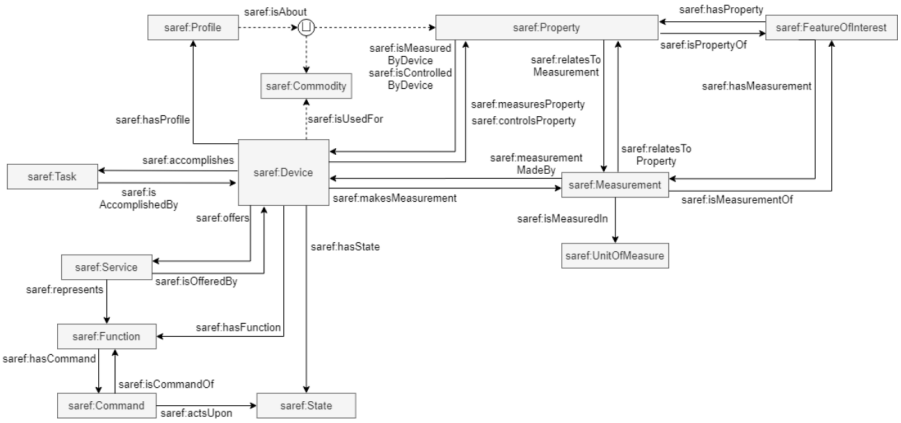


Fig. 7: Overview of the SAREF ontology

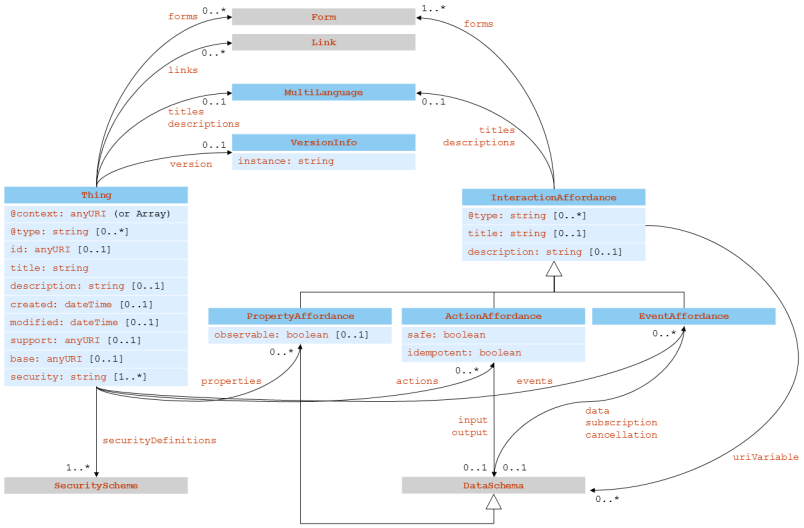


Fig. 8: WoT TD core vocabulary

The WoT TD model is based on four main vocabularies: (1) the core TD Vocabulary, reflecting WoT's paradigm of Properties, Actions and Events, (2) the Data Schema Vocabulary, including the terms defined by JSON Schema (data types and data validation), (3) the WoT Security Vocabulary, reflecting the security mechanism and associated configuration requirements, and (4) the Hypermedia

Controls Vocabulary, encoding the main principles of RESTful communication using Web links and forms. Figure 8 gives an overview of the WoT TD core vocabulary. In the WoT TD model, Action is the function of the Thing, Property is used for sensing and controlling parameters, Form refers to the manner to access a function, and Links relates things together based on Web link specifications.

Published as a W3C recommendation and as an OGC (Open Geospatial Consortium) implementation standard, SSN (Semantic Sensor Network) and SOSA (Sensor, Observation, Sample, and Actuator) [15] are a set of ontologies that describe sensors, actuators, samplers as well as their observations, actuation, and sampling activities. The set of ontologies adopts a modular architecture with SOSA as a self-contained core that is extended by SSN and other modules to add expressivity and breadth. SSN and SOSA mainly cover the physical aspect of the IoT world (including sensors, actuators and samplers) and the modeling of the corresponding data, as shown in Figure 9. However, the definition of the interfaces and services (or operations) of such devices is not included.

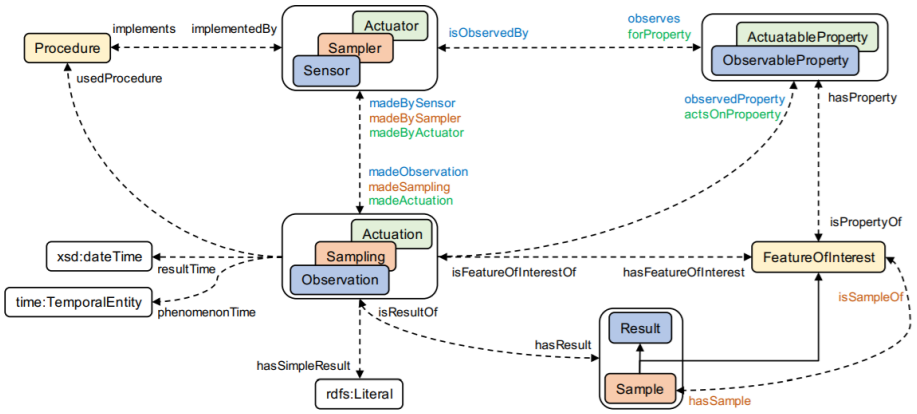
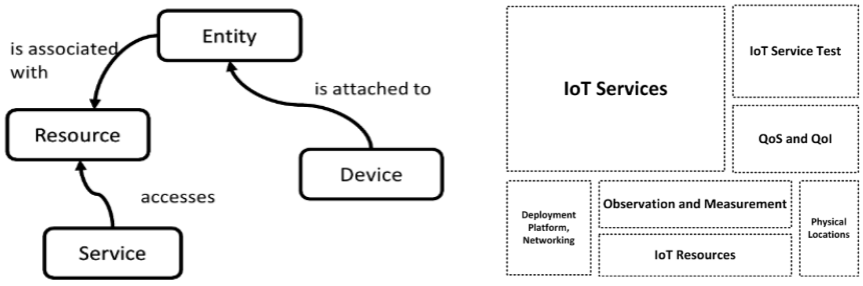


Fig. 9: Overview of the SOSA/SSN ontology

The IoT-A model [9], presented in Figure 10(a), defines the main concepts of the domain IoT and describes the relationships between them. The concepts are the Entity, the Device, the Resource and the Service. Entity refers to "thing" in the Internet of Things and can be a human, animal or automobile object. The Device allows the entity to be part of the digital world by representing interactions. The resource represents an actual software component that provides information about the Entity or controls the Device. The Service provides a well-defined and standardized interface, offering the necessary functionalities to interact with entities. IoT-A therefore provides an architectural basis for other IoT projects.

Wang model [27] is a semantic description ontology for the representation of knowledge in the IoT domain. Figure 10(b) presents an overview of the seven



(a) Key concepts of the IoT-A model (b) Modules of the Wang model

Fig. 10: IoT-A and Wang ontology models

modules of the ontology, namely: (1) the IoT Services module, which exposes the functionalities of the resources hosted on the devices, (2) the IoT Resources module that extends the SSN ontology by including other important resources in the IoT domain such as actuators, (3) the Observations and Measurements module, which allows to describe the real data generated, (4) the Physical Locations module, which includes concepts used for the lookup and discovery of IoT devices, (5) the Deployment Platform Networking module, which provides descriptions on how the IoT resources are organised and deployed as well as the system they form, (6) the Quality of Services and the Quality of Information module that includes important concepts used in many fields, in particularly in the composition and the adaptation of services for the providers and consumers of IoT services, (7) the IoT Service Test module that allows testing and verifying the functional and non-functional capabilities of IoT services during the design and deployment stages.

ForwardDS-IoT [12] is a semantic description model based on existing ontologies for the semantic description of objects using the SSN and SAN ontologies, their location by referring to the "Basic Geo" vocabulary[13], which defines concepts such as the latitude, longitude and altitude of a geographic location, and IoT services using the OWL-S ontology[14]. Figure 11 gives an overview of the ForwardDS-IoT semantic model.

SSN is an ontology that can be considered "too heavy" for a dynamic environment because of the large number of concepts it offers and which are often not used. IoT-Lite [3] is an instantiation of the SSN ontology to describe key IoT concepts enabling interoperability and sensory data discovery in heterogeneous IoT platforms through light semantics. Figure 12 illustrates the key concepts of IoT-Lite model related to three main classes: Object, System and Service, the latter being slightly described.

[13] https://www.w3.org/2005/Incubator/geo/XGR-geo/
[14] https://www.w3.org/Submission/OWL-S/

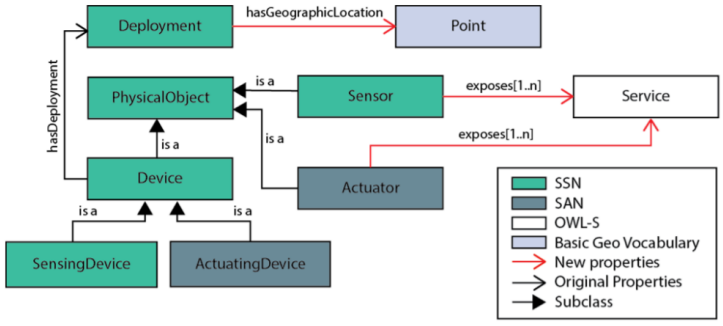


Fig. 11: Overview of the ForwardDS-IoT model

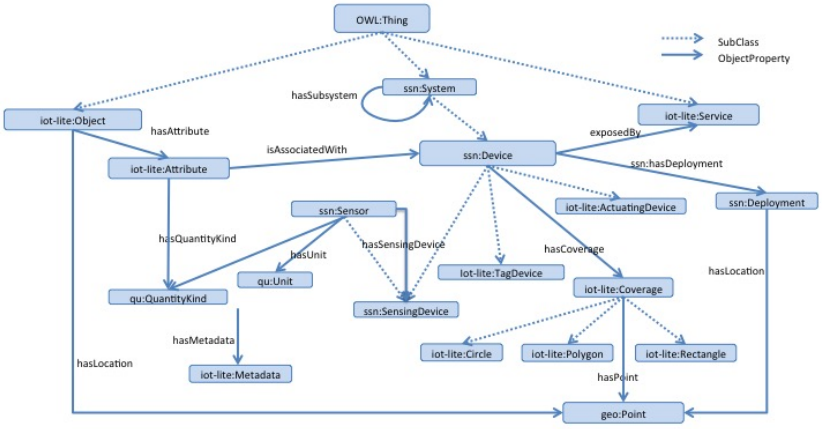


Fig. 12: IoT-Lite ontology

3.2 Evaluation

In table 1, we show the evaluation summary of the existing IoT/WoT-based models previously described, according to the criteria presented in the beginning of Section 3. We used the "+" symbol to express a positive coverage for a criterion, and the "-" symbol to express a lack of a criterion coverage. First, as seen in Table 1, all of the existing models are not thorough. They mainly describe the elementary services, exposed only by devices. Few are the models that include composed services aspects within their descriptions (knowing that these models are SOAP-based and not REST-based), and none of the them consider virtual services that simulate the behavior of real ones. Second, all of the models include the functions provided by the services in their service description, and the majority describe services Input/Output parameters, as well as their location. However, they all lack in defining services links, which is an important criteria that can (1) facilitate service discovery and selection, and (2) ease service replacement whenever a service is no more available. This is apart of the WoT TD model [5], that although it supports services links (to know the services that can be called next to a current services state), the latters are not semantically defined. Third, although quality of services are covered by most of the IoT/WoT models, they are related to two main levels: Physical and Network, with a less attention to the application level.

Table 1: Evaluation of existing IoT/WoT-based models w.r.t. the identified criteria

	Thorough Model						Expressiveness				Service Quality		
	Objets Services	Applications Services	Elementary Services	Composed Services	Virtual Services	Categories	Provided Function	I/O	Location	Links	Physical	Network	Application
IoT-O	+	+	+	-	-	-	+	+	+	-	+	-	-
oneM2M	+	-	+	-	-	-	+	+	-	-	-	-	-
SAREF	+	-	+	-	-	-	+	-	-	-	-	-	-
WoT-TD	+	+	+	-	-	-	+	-	-	+	-	-	-
SSN & SOSA	+	-	+	-	-	-	+	-	+	-	+	+	-
IoT-A	+	-	+	-	-	-	+	-	+	-	-	-	-
Wang Model	+	-	+	+	-	-	+	+	+	-	+	+	+
ForwarDS-IoT	+	-	+	+	-	-	+	+	+	-	+	+	-
IoT-Lite	+	-	+	-	-	-	+	-	+	-	+	+	-

4 Web Resource Model

In order to facilitate the identification, selection, and composition of RESTful-based Web services, i.e., exposed as resources either by WoT connected devices or by Web applications, and allow for their automatic process, we describe, in this section, a dedicated ontology called WoR (Web of Resource). WoR describes Web resources in a normalized way, through a vocabulary that can be used by different Web solutions/platforms.

4.1 WoR Ontology Features

WoR describes Web resources' functional and non-functional aspects, including their composition features (whenever they are composed together in the same resource composition). It also includes resources' visual characteristics, which can facilitate the definition of the composed resources and ease the understanding of their business process (e.g., syntactic and semantic linking, hierarchical representations, etc.). WoR extends several ontologies:

- **HSSN**: it is an extension of the SSN ontology that is originally used to represent sensors, actuation information, and data observation and measurement patterns. HSSN [19] adds to SSN, sensor mobility and multimedia data related concepts, in order to have a representation of hybrid sensor networks, i.e., networks containing mobile/stationary sensors, scalar/multimedia properties, and infrastructures/devices as platforms where sensors are deployed.
- **SOSA**: it is used to model the interaction between the entities involved in the acts of observation, actuation, and sampling [15].

Figure 13 shows the integration and extension of these ontologies in WoR.

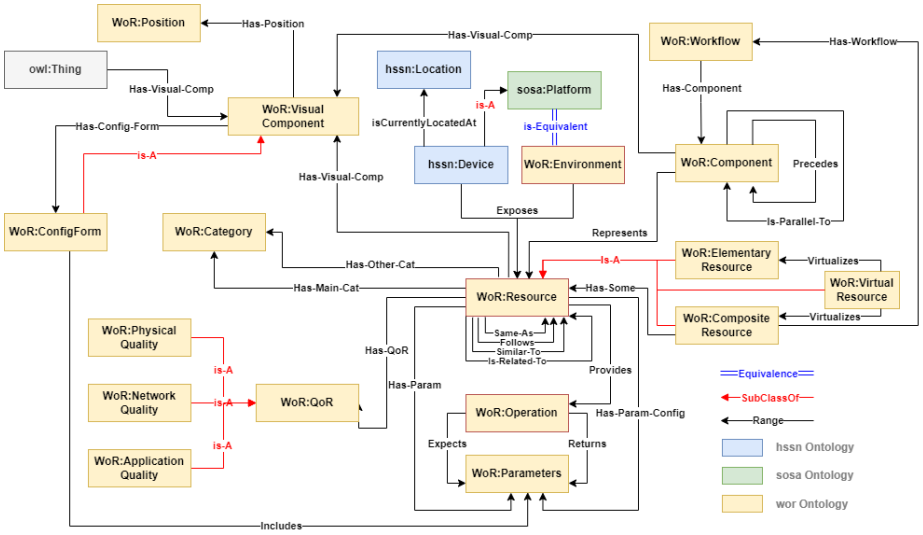


Fig. 13: Overview of the proposed Web resource model

4.2 WoR Ontology Extensions

As shown in Figure 13, WoR extends the existing model ontologies by proposing new concepts and relations, while remaining compliant with existing standards. The extensions are related to:

(i) The thoroughness feature, by defining the "Exposes" relation that links *hssn:Device* and *sosa:Platform* concepts to *WoR:Resource* entity, allowing resources to be exposed by devices and application platforms. The concept *WoR:Resource* is defined as an RDFS[15] resource type. This is inspired from Hydra vocabulary that is used to describe RESTful services by leveraging the power of Linked Data [18]. In addition, each WoR resource, which can be related to specific categories (*WoR:Category*), can be either an elementary resource (*WoR:ElementaryResource*), or a virtual resource (*WoR:VirtualResource*), or a composite resource (*WoR:CompositeResource*). Each composite resource has a workflow, *WoR:Workflow*, which is formed by several components, *WoR:Component*;

(ii) The expressiveness feature, by defining reflexive semantic links between the resources, in addition to the resources provided function, defined in *WoR:Operation*, their I/O parameters, *WoR:Parameters*, and their location using *hssn:Location* (whenever they are exposed by connected devices);

(iii) The resources quality aspects, by defining the *WoR:QoR* entity, which can be either *WoR:PhysicalQuality*, or *WoR:NetworkQuality*, or *WoR:ApplicationQuality*;

(iv) The visual characteristics of resources, allowing to facilitate the understanding of the behavior of linked resources, their required inputs parameters, etc. This done by defining the concepts *WoR:Position*, *WoR:VisualComponent*, and *WoR:ConfigForm*.

Extensions related to the thoroughness resource model feature: As shown in Section 3, most of the IoT/WoT-based models focus on the resources that are only exposed by connected devices, without considering the resources published by Web applications. Therefore, and in order to allow the discovery, the selection, and the composition of all kind of resources, we extended WoR by adding the "Exposes" relation between each of the *hssn:Device* and *sosa:Platform* entities (with the latter being equivalent to the *WoR:Environment*), and *WoR:Resource* concept, denoting that a resource can be exposed either by a device or by an application platform. Resources can be grouped in different categories, e.g., "Data collection", "Data pre-processing", "Advanced data processing", etc. Thus, each resource can be linked to a category (*WoR:Category*), allowing to organize the connected resources, and to facilitate the understanding of their Web related environment.

Moreover, in some cases, there is no single resource that can answer user request. However, combining two or more resources together in the same scenario, referred to as a resource composition, can generate the required results. To form a resource composition, resource discovery and selection are necessary, before being executed by a dedicated orchestration process [1]. Once formed, and in order to avoid re-executing both resource discovery and selection, which consume several resources (e.g., CPU and memory) and time, it is important

[15] Resource Description Framework Schema: `https://www.w3.org/TR/rdf-schema/`

to store the composed resource and model it with the rest of elementary resources. For this aim, we distinguish in WoR between an elementary resource (*WoR:ElementaryResource*) and a composite resource (*WoR:CompositeResource*). The latter is formed by several other resources (composite or/and elementary ones). In order to allow the re-execution of a stored composed resource, we added the *WoR:Worflow* entity to detail the sequence process of the resources forming each composite resource. Such process is described by at least 2 components (*WoR:Component*) representing the resources included within the resource composition. The order of the resources is defined by the defined relations "Precedes" and "Is-Parallel-To".

Also, in order to allow to mimick the behavior of missing real resources, e.g., the resources exposed by physical devices that are very costly to use, we allow in WoR to simulate such behavior, by defining virtual resources through the *WoR:VirtualResource* entity, a sub-class of the *WoR:Resource* that can virtualize both the *WoR:ElementaryResource* and the *WoR:CompositeResource* entities.

Extensions related to the expressiveness resource model feature: Hydra model, from which we were inspired to define a Web resource as an RDFS type resource, is a vocabulary that specifies a number of concepts that are used to describe resources, while enabling a server to advertise valid state transitions to a client. Within this context, Hydra defines the Link concept that enables the dynamic discovery of the next resources that can be called next at runtime. However, there are no semantic data about the type of the Link (e.g., whether it is complementary or similar for instance), which can facilitate the automatic resource discovery, selection, and composition, as well as the replacement of a resource in case it is no more available. Therefore, we extended WoR by defining reflexive semantic relations between the resources:

- "Same-As" link, denotes that the related resources provide the exact same function
- "Follows" link, denotes that the related resources can be executed in a complementary manner based on the dependency of their provided function
- "Similar-To" link, indicating that the related resources provide a similar function
- "Is-Related-To" link, indicating that the related resources are exposed by devices installed in the same zone/location

The expressiveness model feature is also covered by defining the provided function attribute to the *WoR:Operation* of the exposed resources, with its input parameters (through the "Expects" relation between *WoR:Operation* and *WoR:Parameter*, its output parameters (through the "Returns" relation between *WoR:Operation* and *WoR:Parameter*), as well as resources location (*hssn:Location*) whenever they are exposed by devices. We note that in our model, the *WoR:Operation* represents the information necessary for clients to construct valid HTTP requests in order to call/manipulate the resource. As such, each

WoR:Operation consists of a required HTTP method, optional input and output parameters, and information about the function provided by each resource Operation, such as "Collect temperature", or "Predict energy consumption", etc.

Extensions related to resources quality aspects: In many cases, there are several resources that can be discovered having the same function. In order to distinguish between such resources, and select the most convenient ones answering user request, we defined, in WoR model, quality of resources (*WoR:QoR*), which can be divided into 3 main groups: (1) Physical quality (*WoR:PhysicalQuality*), representing the aspects that describe the quality of the IoT/WoT devices exposing the resources (e.g., Battery and Operation Range), (2) Network quality (*WoR:NetworkQuality*), denoting the aspects that specify the quality of the data transferred between the resources (e.g., Bandwidth and Latency) and (3) Application quality (*WoR:ApplicationQuality*), representing the quality of the service provided by the resources (e.g., Response Time, Availability, etc.).

Extensions related to resource visual characteristics: In order to facilitate the definition of composed resources and ease the understanding of their business process (e.g., syntactic and semantic linking, hierarchical representations, etc.), we added in WoR several concepts to represent visually a resource (elementary, composed, or virtual) with its characteristics. To do so, we use *WoR:ConfigForm* to represent visually the configuration form of a resource, which includes configurable parameters necessary to execute the resource (e.g., the "K" parameter related to the resource that applies the KNN (K-nearest neighbors) clustering algorithm. We also added *WoR:VisualComponent* to represent visually each resource with an attached icon (i.e., an image). As for the *WoR:Position* entity, it is used to describe the position of a resource on the user interface.

5 Experimental Evaluation

In this section, we present the experimental protocol that we followed in order to evaluate the WoR ontology, on both syntactic and semantic aspects. It is based on 4 parts of evaluation:

1. *Accuracy Evaluation:* In which we aim to verify if the concepts/properties defined in the ontology are able to meet the different objectives, presented in Section 5.1 below, and to cover the criteria explained in Section 3.
2. *Clarity Evaluation:* In which we seek to verify whether the names or labels used to describe the concepts/properties are clear, and that are not ambiguous for users (experts and non-experts).
3. *Performance Evaluation:* In which we aim to study the response time of different simple and complex queries, with the evolution of the resources graph based on several variations (e.g., increasing the number of devices exposing resources, and increasing the number of resources and functions).

4. *Consistency Evaluation:* This part is used to check if the added concepts/ properties generate inconsistencies within the ontology structure (e.g., check if there are concepts that do not have no parents).

5.1 Accuracy Evaluation

In this part, we define the most useful queries that cover our objectives in terms of: (i) Exploration, to search for the set of the resources that are connected, and to have a better understating of the Web environment, (ii) Discovery, to identify the necessary resources that respond to user demands, (iii) Selection, to select among the candidate resources (the ones providing the same required functions) the best ones answering user demands, and (iv) Composition/Execution, to link the identified and selected resources together, forming a composition, and execute them according to their corresponding order. Moreover, we analyze these queries according to their ability in supporting the necessary criteria presented in Section 3.

In Table 2, we present the set of the queries that we find useful in covering the required objectives and the necessary criteria. The queries are defined and expressed in SPARQL[16] in Appendix A.

Table 2: List of useful queries covering the required objectives and criteria

	Query	Objectives				Criteria		
		Exploration	Discovery	Selection	Composition/ Execution	Thorough Model	Expressiveness	Resource Quality
A	Retrieve different types of Web resources	+	-	-	-	+	-	-
B	Retrieve the Web resources providing a given function	-	+	+	-	-	+	-
C	Retrieve the list of all the functions provided by the Web environment	+	-	-	-	-	+	-
D	Retrieve the output parameters of a Web resource, and the input parameters of another	-	-	+	-	-	+	-
E	Retrieve the Web resources exposed in a given location	+	+	+	-	-	+	-
F	Retrieve the Web resources that are the same (same-as) as a Web resource	-	+	+	-	-	+	-
G	Retrieve the Web resources that are complementary (follows) to a Web resource	-	+	+	-	-	+	-
H	Retrieve the workflow of a composed Web resource	-	-	-	+	+	-	-
I	Retrieve the sequential order of the workflow components of a composed Web resource	-	-		+	+	-	-
J	Retrieve the functions provided by the virtual Web resources	+	-	-	-	+	+	-
K	Retrieve a Web resource providing a given function with a quality criterion (e.g., Accuracy >80%)	-	+	+	-	-	+	+
L	Retrieve a Web resource collecting data within a location with a quality criterion (e.g. Bandwidth>400 Mbits/sec)	-	+	+	-	-	+	+

5.2 Clarity Evaluation

In order to evaluate the ambiguity of the labels used to describe some of the newly defined WoR concepts and their object properties, we created a questionnaire consisting of 14 multiple choice questions, in which we have proposed

[16] A standard query language and protocol for Linked Open Data on the Web, that is able to retrieve and manipulate data stored in Resource Description Framework (RDF) format.

several names alternatives for the defined concepts and properties. The questionnaire was filled by 20 male and female participants (i.e., Technical Designers, Functional Designers, and R&D Engineers/Experts). The participants have been asked to choose the best name, from a list of 3 choices synonyms in the domain, that fit each new concept/property, and if needed, they could suggest other concepts/properties which they find more suitable to use. The questionnaire was divided into two parts: A part where we proposed several names alternatives for some of the WoR concepts, and another part where we proposed several names alternatives for some of the object properties linking concepts together. Figure... shows that the terms used in the first part (for the WoR concepts) are clear for the participants with an average of ... Figure... shows that labels used in the second part (for the WoR object properties) were comprehensible to the participants with an average of ....

5.3 Performance Evaluation

In order to evaluate the performance of the WoR ontology, we considered several scenarios to study the impact of the evolution of the resources' graph on WoR's performance. The performance is based on the execution of several queries (picked from the list of queries defined in Section 5.1) in different scenarios consisting on different resources' graph settings: (1) varying the number of resources with their provided functions, (2) varying the number of devices exposing data collection resources and their distribution into different number of locations, (3) varying the number of involved resource in a composition. In the experiments, we show the query response time (ms) based on the average of 10 sequential executions for each query. The tests were conducted on a Windows 10 Professional machine with an Intel i7-8665U CPU @ 1.90GHz 2.11GHz processor and 1 GB RAM, using Stardog (`https://www.stardog.com/`), an Enterprise Knowledge Graph platform and graph DBMS with high availability, high performance reasoning, and virtualization.

Resource and Function Impact.

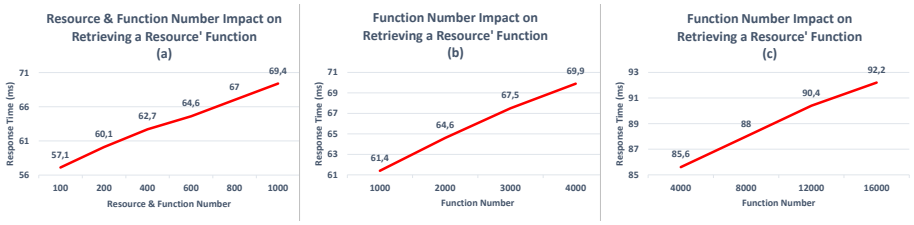


Fig. 14: Resource and Function number impact

In the first scenario (see Figure 14), we studied the impact of varying the number of exposed resources with the number of their provided functions. In

Figure 14-(a), the number of resources is equivalent to the number of functions, where each resource provides one single function. In Figure 14-(b) and Figure 14-(c), we fixed the number of resources, respectively, on 500 and 4000, and increased the number of functions. In the latters, a resource provides more than one function. In each of these resources graph setups, we retrieved the list of resources that provide a specific given function by applying the query B (see Table 2) and measured its corresponding response time. The increase of the query run-time in all of the graphs presented in Figure 14 is explained by the number of additional resources and functions expanding the search environment. However, we can see that the number of resources has more impact than the number of functions, as the response time is almost the same (64,4 ms and 64,9 ms) when having 1000 resources and 1000 functions (Figure 14-(a)) versus 500 resources and 4000 functions (Figure 14-(b)).

Device and Location Impact. In the second scenario (see Figure 15-(a) and Figure 15-(b)), we studied the impact of varying the number of devices exposing data collection resources and their distribution ratio in different numbers of locations. In Figure 15-(a), we increased the number of devices exposing Web resources and distributed them, equally (i.e., each location contained the exact same number of devices), on a fixed number of locations (50 locations). In these tests, the distribution ratio, which is equal to the number of devices divided by the number of locations, was thus evolving. In Figure 15-(b), we fixed the number of devices exposing resources (1000 devices) and distributed them, equally, on a varied number of locations. In such cases, the distribution ratio was decreasing. For configuration set in Figures 15-(a) and 15-(b), we retrieved the list of resources that are exposed in a specific given location by applying the query E (see Table 2) and measured its corresponding response time. The increase of the query run-time in the graphs is explained by the number of additional devices and locations expanding the search environment. We can also observe that the evolving number of locations in which the devices are distributed has more influence than increasing the number of devices. This can be seen from the response time that has risen more (from 59,1 ms to 72,4 ms) when multiplying the number of locations by 10 (from 50 to 500), comparing to multiplying the number of devices by 10 (from 100 to 1000) where the response time increased from 59,1 ms to 68,1 ms.

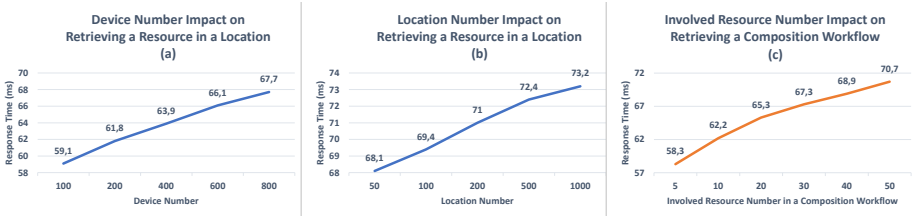


Fig. 15: Device and Location impact vs Involved Resources in a Composition Workflow impact

Involved Resources in a Composition Workflow Impact. In the third scenario (see Figure 15-(c)), we studied the impact of increasing the number of resources involved in a composition workflow. In this scenario, we retrieved the list of resources that form a compose Web resource by applying the query H (see Table 2) and measured its corresponding response time. As shown in Figure 15-(c), the response time evolves almost linearly with the increasing number of resources, as there are more resources to get.

Discussion. The generated graphs in all of the different experimental scenarios show a promising and positive linear curve, denoting that the query response time increases linearly with the increasing number of resources, their provided functions, the number of connected devices (exposing resources) and their distribution into many locations, as well as the number of resources forming a resource composition. This indicates a proportional relation, which is a quasi-constant increase between the different variables used and the response time of the queries. The results also highlight that the growing number of resources and the distribution of the connected devices in many locations, has more impact on the increase of the queries response time, comparing to the other variables.

6 Conclusion

In this paper, we propose a Web of Resources ontology (WoR), that is used to describe the services (the resources) provided by the devices and/or applications exposed by connected Web environments. It includes the functional and non-functional aspects of the resources, as well as their composition features (whenever they are composed together), and some visual characteristics to facilitate their composition and ease the understanding of their process. WoR mainly reuses several existing known IoT-based models (i.e., HSSN, an extension of the Semantic Sensor Network ontology (SSN), which adds sensor mobility and multimedia data related concepts, and SOSA (Sensor, Observation, Sample, and Actuator ontology), which describes sensors, actuators, samplers as well as their observations, actuation, and sampling activities. We implemented WoR, and evaluated its accuracy and performance in several scenarios that have shown promising results. As future work, we seek to continue the evaluation of the ontology to test its clarity and consistency. Finally, we want to use the proposed solution in the Web platform offered by OpenCEMS, and in ongoing Web-based projects provided by Open Group ().

Appendix A Useful Queries Expressed in SPARQL Covering the Required Objectives and Criteria

Query A1: Retrieve the list of all Web resources

```
SELECT ?Res_id ?Res_title
WHERE {
        ?res rdf:type gradys:Resource; gradys:Id ?Res_id;
        gradys:Title ?Res_title; gradys:Exposed_by ?plat .
        ?plat rdf:type gradys:Platform
      }
```

Query A2: Retrieve the Web resources exposed by connected objects

```
SELECT ?Res_id ?Res_title
WHERE {
        ?res rdf:type gradys:Resource; gradys:Id ?Res_id ;
        gradys:Title ?Res_title; gradys:Exposed_by ?dev .
        ?dev rdf:type gradys:Device
      }
```

Query A3: Retrieve the Web resources exposed by connected objects

```
SELECT ?Res_id ?Res_title
WHERE {
        ?res rdf:type gradys:Resource; gradys:Id ?Res_id ;
        gradys:Title ?Res_title; gradys:Exposed_by ?plat .
        ?plat rdf:type gradys:Platform
        FILTER NOT EXISTS {
                            ?res gradys:Exposed_by ?dev .
                            ?dev rdf:type gradys:Device
                          }
       }
```

Query A4: Retrieve the list of elementary Web resources

```
SELECT ?Res_id ?Res_title
WHERE {
        ?res rdf:type gradys:ElementaryResource;
        gradys:Id ?Res_id; gradys:Title ?Res_title
      }
```

Query A5: Retrieve the composed Web resources exposed by platforms (which are not connected objects)

```
SELECT ?Res_id ?Res_title
WHERE {
       ?res rdf:type gradys:CompositeResource;
       gradys:Id ?Res_id; gradys:Title ?Res_title;
       gradys:Exposed_by ?plat .
       ?plat rdf:type gradys:Platform
       FILTER NOT EXISTS {
                           ?res gradys:Exposed_by ?dev .
                           ?dev rdf:type gradys:Device
                         }
      }
```

Query A6: Retrieve the Web resources belonging to a given category

```
SELECT ?Res_id ?Res_title
WHERE {
        ?res rdf:type gradys:Resource .
        ?res gradys:Has_MainCategory ?cat .
        ?cat gradys:Category_value "Data Collection"^^xsd:string .
        ?res gradys:Id ?Res_id . ?res gradys:Title ?Res_title
      }
```

Query A7: Retrieve the Web resources providing a given function

```
SELECT ?Res_id ?Res_title
WHERE {
        ?res rdf:type gradys:Resource;
        gradys:Provides ?Operation .
        ?Operation gradys:Function "Collect Temperature" .
        ?res gradys:Id ?Res_id; gradys:Title ?Res_title
      }
```

Query A8: Retrieve the list of all the functions provided by the Web environment

```
SELECT ?fun
WHERE {
        ?res rdf:type gradys:Resource.
        ?res gradys:Provides ?Operation .
        ?Operation gradys:Function ?fun
      }
```

Query A9: Retrieve the output parameters of a Web resource and the input parameters of another

```
SELECT ?res1id ?res2id ?Paramin_name ?Paramin_datatype
?Paramout_name ?Paramout_datatype
WHERE {
        {
            ?res1 rdf:type gradys:Resource .
            ?res1 gradys:Id "Res_4"^^xsd:string .
            ?res1 gradys:Id ?res1id .
            ?res1 gradys:Provides ?op .
            ?op gradys:Expects ?Paramin .
            ?Paramin gradys:Param_name ?Paramin_name .
            ?Paramin gradys:Param_datatype ?Paramin_datatype
        }
        UNION
         {
            ?res2 rdf:type gradys:Resource .
            ?res2 gradys:Id "Res_2"^^xsd:string .
            ?res2 gradys:Id ?res2id .
            ?res2 gradys:Provides ?op .
            ?op gradys:Returns ?Paramout .
            ?Paramout gradys:Param_name ?Paramout_name .
            ?Paramout gradys:Param_datatype ?Paramout_datatype
          }
       }
```

Query A10: Retrieve the Web resources exposed in a given location

```
SELECT ?Res_id ?Res_title
WHERE {
        ?res rdf:type gradys:Resource .
        ?res gradys:Id ?Res_id .
        ?res gradys:Title ?Res_title .
        ?dev rdf:type hssn:Device .
        ?dev gradys:Exposes ?res .
        ?loc rdf:type hssn:Location .
        ?dev hssn:currentlyLocatedAt ?loc .
        ?loc gradys:Location_id "Zone 1"^^xsd:string
      }
```

Query A11: Retrieve the Web resources that are the same (same-as) as a Web resource

```
SELECT ?res_id
WHERE {
        ?res rdf:type gradys:Resource .
        ?res gradys:Id "Res_5"^^xsd:string .
        ?res gradys:Same_As ?otheres .
        ?otheres gradys:Id ?res_id
      }
```

Query A12: Retrieve the Web resources that are complementary (follows) to a Web resource

```
SELECT ?res_id
WHERE {
        ?res rdf:type gradys:Resource .
        ?res gradys:Id "Res_3"^^xsd:string .
        ?res gradys:Follows ?otheres .
        ?otheres gradys:Id ?res_id
      }
```

Query A13: Retrieve the workflow of a composed Web resource

```
SELECT ?list_res_id
WHERE {
        ?res rdf:type gradys:CompositeResource .
        ?res gradys:Id "Res_co_1"^^xsd:string .
        ?res gradys:Has_Workflow ?wf .
        ?wf gradys:Has_Component ?comp .
        ?comp gradys:Represents ?list_res .
        ?list_res gradys:Id ?list_res_id
      }
```

Query A14: Retrieve the sequential order of the workflow components of a composed Web resource

```
SELECT ?list_res1_id ?list_res2_id
WHERE {
        ?res1 rdf:type gradys:CompositeResource .
        ?res1 gradys:Id "Res_co_1"^^xsd:string .
        ?res1 gradys:Has_Workflow ?wf .
        ?wf gradys:Has_Component ?comp .
        ?comp gradys:Represents ?list_res1 .
        ?list_res1 gradys:Id ?list_res1_id .
        ?comp gradys:Precedes ?othercomp .
        ?othercomp gradys:Represents ?list_res2 .
        ?list_res2 gradys:Id ?list_res2_id
      }
```

Query A15: Retrieve the functions provided by the virtual Web resources

```
SELECT ?fun
WHERE {
        ?res rdf:type gradys:VirtualResource.
        ?res gradys:Provides ?Operation .
        ?Operation gradys:Function ?fun
      }
```

Query A16: Retrieve a Web resource providing a given function with a quality criterion (e.g. Accuracy $>$ %80)

```
SELECT ?Res_id ?Res_title
WHERE {
        ?res rdf:type gradys:Resource.
        ?res gradys:Provides ?Operation .
        ?Operation gradys:Function "Temperature Prediction" .
        ?res gradys:Id ?Res_id .
        ?res gradys:Title ?Res_title .
        ?res gradys:Has_QoR ?qor .
        ?qor gradys:QoR_name "Accuracy"^^xsd:string .
        ?qor gradys:QoR_value ?val .
        FILTER (?val > 80)
       }
```

Query A17: Retrieve a Web resource collecting data within a location with a quality criterion (e.g. Bandwidth> 400 Mbits/sec)

```
SELECT ?Res_id ?Res_title
WHERE {
        ?res rdf:type gradys:Resource.
        ?res gradys:Has_MainCategory ?cat .
        ?cat gradys:Category_value "Data Collection"^^xsd:string .
        ?res gradys:Id ?Res_id .
        ?res gradys:Title ?Res_title .
        ?dev rdf:type hssn:Device .
        ?dev gradys:Exposes ?res .
        ?loc rdf:type hssn:Location .
        ?dev hssn:currentlyLocatedAt ?loc .
        ?loc gradys:Location_id "Zone 1"^^xsd:string .
        ?res gradys:Has_QoR ?qor .
        ?qor gradys:QoR_name "Bandwidth"^^xsd:string .
        ?qor gradys:QoR_value ?val .
        FILTER (?val > 400)
      }
```

References

1. U Arul and S Prakash. Toward automatic web service composition based on multilevel workflow orchestration and semantic web service discovery. *International Journal of Business Information Systems*, 34(1):128–156, 2020.
2. Payam Barnaghi, Amit Sheth, and Cory Henson. From data to actionable knowledge: Big data challenges in the web of things [guest editors' introduction]. *IEEE Intelligent Systems*, 28(6):6–11, 2013.
3. Maria Bermudez-Edo, Tarek Elsaleh, Payam Barnaghi, and Kerry Taylor. Iot-lite: a lightweight semantic model for the internet of things and its use with dynamic semantics. *Personal and Ubiquitous Computing*, 21(3):475–487, 2017.
4. Dario Bonino, Fulvio Corno, and Luigi De Russis. Poweront: An ontology-based approach for power consumption estimation in smart homes. In *International Internet of Things Summit*, pages 3–8. Springer, 2014.
5. Victor Charpenay and Sebastian Käbisch. On modeling the physical world as a collection of things: The w3c thing description ontology. In *European Semantic Web Conference*, pages 599–615. Springer, 2020.
6. Michael Compton, Payam Barnaghi, Luis Bermudez, Raul Garcia-Castro, Oscar Corcho, Simon Cox, John Graybeal, Manfred Hauswirth, Cory Henson, Arthur Herzog, et al. The ssn ontology of the w3c semantic sensor network incubator group. *Journal of Web Semantics*, 17:25–32, 2012.
7. Laura Daniele, Frank den Hartog, and Jasper Roes. Created in close interaction with the industry: the smart appliances reference (saref) ontology. In *International Workshop Formal Ontologies Meet Industries*, pages 100–112. Springer, 2015.
8. Soumya Kanti Datta and Christian Bonnet. Extending datatweet iot architecture for virtual iot devices. In *2017 IEEE International Conference on Internet of*

Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData), pages 689–694. IEEE, 2017.

9. Suparna De, Payam Barnaghi, Martin Bauer, and Stefan Meissner. Service modelling for the internet of things. In *2011 Federated Conference on Computer Science and Information Systems (FedCSIS)*, pages 949–955. IEEE, 2011.
10. F De Carvalho Diniz. Composition of semantically enabled geospatial web services. Master's thesis, University of Twente, 2016.
11. Roy T Fielding, Richard N Taylor, Justin R Erenkrantz, Michael M Gorlick, Jim Whitehead, Rohit Khare, and Peyman Oreizy. Reflections on the rest architectural style and" principled design of the modern web architecture"(impact paper award). In *Proceedings of the 2017 11th Joint Meeting on Foundations of Software Engineering*, pages 4–14, 2017.
12. Porfirio Gomes, Everton Cavalcante, Taniro Rodrigues, Thais Batista, Flavia C Delicato, and Paulo F Pires. A federated discovery service for the internet of things. In *Proceedings of the 2nd Workshop on Middleware for Context-Aware Applications in the IoT*, pages 25–30, 2015.
13. Amelie Gyrard, Soumya Kanti Datta, and Christian Bonnet. A survey and analysis of ontology-based software tools for semantic interoperability in iot and wot landscapes. In *2018 IEEE 4th World Forum on Internet of Things (WF-IoT)*, pages 86–91. IEEE, 2018.
14. Armin Haller, Krzysztof Janowicz, Simon JD Cox, Maxime Lefrançois, Kerry Taylor, Danh Le Phuoc, Joshua Lieberman, Raúl García-Castro, Rob Atkinson, and Claus Stadler. The modular ssn ontology: A joint w3c and ogc standard specifying the semantics of sensors, observations, sampling, and actuation. *Semantic Web*, 10(1):9–32, 2019.
15. Armin Haller, Krzysztof Janowicz, Simon JD Cox, Maxime Lefrançois, Kerry Taylor, Danh Le Phuoc, Joshua Lieberman, Raúl García-Castro, Rob Atkinson, and Claus Stadler. The sosa/ssn ontology: A joint w3c and ogc standard specifying the semantics of sensors, observations, actuation, and sampling. *Semantic Web-Interoperability, Usability, Applicability an IOS Press Journal*, 56:1–19, 2019.
16. Simon B Heilesen. A short history of designing for communication on the web. In *Designing for Networked Communications: Strategies and Development*, pages 118–136. IGI Global, 2007.
17. Karwan Jacksi and Shakir M Abass. Development history of the world wide web. *Int. J. Sci. Technol. Res*, 8(9):75–79, 2019.
18. Markus Lanthaler and Christian Gütl. Hydra: A vocabulary for hypermedia-driven web apis. In *LDOW*, 2013.
19. Elio Mansour, Richard Chbeir, and Philippe Arnould. Hssn: an ontology for hybrid semantic sensor networks. In *Proceedings of the 23rd International Database Applications & Engineering Symposium*, pages 1–10, 2019.
20. Sanju Mishra and Sarika Jain. Ontologies as a semantic model in iot. *International Journal of Computers and Applications*, 42(3):233–243, 2020.
21. Snehal Mumbaikar, Puja Padiya, et al. Web services based on soap and rest principles. *International Journal of Scientific and Research Publications*, 3(5):1–4, 2013.
22. Yusuf Perwej, Kashiful Haq, Firoj Parwej, M Mumdouh, and Mohamed Hassan. The internet of things (iot) and its application domains. *International Journal of Computer Applications*, 975(8887):182, 2019.

23. Jinghai Rao and Xiaomeng Su. A survey of automated web service composition methods. In *International Workshop on Semantic Web Services and Web Process Composition*, pages 43–54. Springer, 2004.
24. Nicolas Seydoux, Khalil Drira, Nathalie Hernandez, and Thierry Monteil. Iot-o, a core-domain iot ontology to represent connected devices networks. In *European Knowledge Acquisition Workshop*, pages 561–576. Springer, 2016.
25. Juris Tihomirovs and Jānis Grabis. Comparison of soap and rest based web services using software evaluation metrics. *Information Technology & Management Science (Sciendo)*, 19(1), 2016.
26. Chen Wang, Hui Ma, Gang Chen, and Sven Hartmann. Evolutionary multitasking for semantic web service composition. In *2019 IEEE Congress on Evolutionary Computation (CEC)*, pages 2490–2497. IEEE, 2019.
27. Wei Wang, Suparna De, Gilbert Cassar, and Klaus Moessner. Knowledge representation in the internet of things: semantic modelling and its applications. *automatika*, 54(4):388–400, 2013.