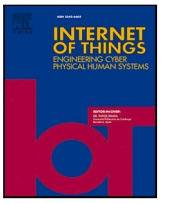
Internet of Things 23 (2023) 100855

Contents lists available at ScienceDirect

Internet of Things

journal homepage: www.elsevier.com/locate/iot



Research article

Critical infrastructure awareness based on IoT context data

Marc Vila [a,b,*], Maria-Ribera Sancho [a,c], Ernest Teniente [a], Xavier Vilajosana [b,d]

[a] InLab FIB, Universitat Politècnica de Catalunya, Barcelona, Spain
[b] Worldsensing, Barcelona, Spain
[c] Barcelona Supercomputing Center, Barcelona, Spain
[d] Universitat Oberta de Catalunya, Barcelona, Spain

ARTICLE INFO

Keywords:
Interoperability
Context-awareness
Semantics
Cloud continuum
Internet of things

ABSTRACT

The Internet of Things (IoT) represents a powerful new paradigm for connecting and communicating with the world around us. It has the potential to transform the way we live, work, and interact with our surroundings. IoT devices are transmitting information over the Internet, most of them with different data formats, despite they may be communicating similar concepts. This often leads to data incompatibilities and makes it difficult to extract the knowledge underlying that data. Because of the heterogeneity of IoT devices and data, interoperability is a challenge, and efforts are underway to overcome this through research and standardization. While data collection and monitoring in IoT systems are becoming more prevalent, contextualizing the data and taking appropriate actions to address issues in the monitored environment is still an ongoing concern. Context Awareness is a highly relevant topic in IoT, as it aims to provide a deeper understanding of the data collected and enable more informed decision-making.

In this paper, we propose a semantic ontology designed to monitor global entities in the IoT. By leveraging semantic definitions, it enables end-users to model the entire process from detection to action, including context-aware rules for taking appropriate actions. The advantages of using semantic definitions include more accurate and consistent data interpretation, which improves the overall monitoring process and enables more effective decision-making based on the collected insights. Our proposal includes semantic models for defining the entities responsible for monitoring and executing actions, as well as the elements that need to be considered for an effective monitoring process. Additionally, we provide a new definition for the components known as gateways, which enable the connection and communication between devices and the Internet. Finally, we show the benefits of our ontology by applying it to a critical infrastructure domain where a rapid response is vital to prevent accidents and malfunction of the entities.

1. Introduction

With billions of gadgets connected to the Internet and producing enormous volumes of data every day, the Internet of Things (IoT) has established itself as a constant presence in our lives. The IoT is a network of physical devices and entities equipped with sensors and actuators. These sensor-equipped devices can gather and transmit data via the Internet, whereas devices with actuators can modify the state of entities in the physical world and communicate data over the Internet, much like sensors.

* Corresponding author at: InLab FIB, Universitat Politècnica de Catalunya, Barcelona, Spain.
E-mail addresses: marc.vila.gomez@upc.edu, mvila@worldsensing.com (M. Vila), maria.ribera.sancho@upc.edu, maria.ribera@bsc.es (M.-R. Sancho), ernest.teniente@upc.edu (E. Teniente), xvilajosana@worldsensing.com, xvilajosana@uoc.edu (X. Vilajosana).

https://doi.org/10.1016/j.iot.2023.100855
Received 12 April 2023; Received in revised form 20 June 2023; Accepted 22 June 2023
Available online 28 June 2023

IoT sensor devices range from simple sensors that measure temperature or humidity to complex machines that monitor and control entire industrial processes. Data collected by these devices can be used to monitor, understand, and improve processes from home appliances to critical infrastructure. On the other hand, IoT actuator devices are often used to convert a signal or input into a physical action, movement, or to modify the logical state of an entity.

Both sensors and actuators play a fundamental role in the IoT. They allow devices to communicate with each other, as well as the cloud-based systems used to process and analyze the data the devices generate. This communication allows devices to be monitored and controlled in real-time, enabling sophisticated applications that can respond quickly to changing conditions.

Context awareness plays an important role in these systems. IoT systems gain deeper insights from the data they are producing and use this knowledge to create more complex and useful applications by understanding the environment in which the devices operate. Considering the data collected by IoT devices and allowing the system to understand the context of that data, the system can react since it is aware of the situation in which it finds itself. The IoT domain heavily relies on context awareness as it enables systems to gain a more precise understanding of their environment, leading to accurate responses to events [1]. Entities are generally monitored with a common goal: to determine their current health or status. For example, a context-aware system in a smart home can turn off the lights and adjust the temperature when it detects that the occupants have left the room. Similarly, in an industrial setting such as a factory, context awareness can facilitate the monitoring of machinery performance and optimization of the production, considering factors like temperature, humidity, or energy consumption.

In addition to context awareness, communication technologies in the IoT encompass a wide range of options, with significant variations in their usage. One common feature among them is the support for the Cloud Continuum [2], which refers to the emerging paradigm that embodies all computing tiers, from the Cloud to the Thing level, including intermediate tiers such as the Edge or the Fog. In this way, the Computing Continuum represents a holistic approach that integrates *things*, computing resources, and infrastructures to enable seamless data processing, storage, and management, aiming to optimize performance, scalability, and efficiency in IoT systems. The Cloud tier is used for applications needing high computational power, and large centralized data processing; the Fog tier is used in scenarios with low latency, local data processing, and real-time interactions; and the Edge tier when ultra-low latency, high bandwidth, and real-time processing at the data source are required [3,4].

When transmitting information to the Internet, some devices require establishing a link to an entry point to the Internet, a *gateway*. Cellular networks are an example where the link is transparent to the user (i.e., 5G, 4G/LTE, or 2G/GPRS); other technologies have a similar situation, where a link must be established with a particular gateway, and communication is only available through that gateway (i.e., WiFi or Bluetooth); and LPWANs, on the other hand, which transmit information through gateways that are considered as one of many possible data transport media, with no specific link to a particular gateway (i.e., LoRa, NB-IoT, SigFox, or LTE-M). In general, there should also be a capability to check the connection's status, and depending on the type of network, the connection may be device-to-gateway or device-to-network, namely a group of gateways. Therefore, when monitoring in real-time, IoT systems need to consider the communication technologies employed, as well as the current communication status [5].

To fully leverage the potential of the IoT, it is crucial to go beyond mere data collection. Systems must interpret the generated data and derive insights from it. Here, the IoT concept becomes significant, empowering the seamless exchange of information between devices and systems, allowing them to work together irrespective of differences in manufacturing processes, manufacturers, or the technology used for data communication. Despite this, there is still a significant challenge in the IoT research field due to the heterogeneity of IoT devices and data formats. As noted by Elkhodr et al. [6], the various forms of IoT devices and the use of multiple communication protocols make data integration and aggregation from various sources challenging. To overcome the heterogeneity challenge, ongoing efforts are being made to improve interoperability in the IoT [7,8]: organizations such as Ericsson [9] or AIOTI [10] are actively working on this topic; in the academia, Noura et al. [11] propose a taxonomy to classify the levels of interoperability, including *device*, *syntactic*, *network*, *semantic*, and *platform*.

Enabling communication among IoT devices involves the abstraction of their distinctive syntax and data formats to ensure complete interoperability through a shared semantic framework, such as an ontology. An ontology is a conceptual representation of the data, concepts, and properties that are managed within a particular domain. According to Noy and McGuiness [12], ontologies offer several advantages: (1) they establish a common understanding of information structure among software agents, (2) they promote the reuse of domain knowledge, (3) they make domain assumptions explicit, and (4) they enable the analysis of domain knowledge. Leveraging these benefits is crucial to achieving interoperability, as ontologies facilitate semantic interoperability among humans, computers, and systems, and hence achieving communication interoperability between software systems [13,14].

In this paper, we propose an ontology to overcome the limitations mentioned so far regarding interoperability, context awareness, and considering the state of the communication during the system execution. With this purpose, our ontology incorporates all the relevant concepts of the IoT domain, ranging from the more functional ones, like sensors or actuators; to the more administrative ones, to indicate where the nodes are physically and logically located. Our ontology allows also modeling the conditions under which the actuation mechanisms should be triggered and how the system should respond to them. Finally, we propose a novel way to modeling gateways, an essential entity in the IoT domain, including sensor data collected directly from its sensors as well as communication data of the things, received via a gateway.

To facilitate the implementation of our ontology, we also propose an architecture for the IoT that includes world entities, sensors, actuators, gateways, communication technologies, and cloud services. The architecture enables monitoring and actuation in physical objects through IoT devices. It also facilitates data collection, transmission, storage, and processing, leading to insights and intelligent decision-making.

Our work is related to industrial research at Worldsensing,[20] a company that focuses on providing services through the monitoring of industrial environments using IoT devices. The development of a platform able to control and act on its own and third-party industrial IoT devices has become a critical goal for the success of the company, and the need to holistically address all these issues can only be successfully established through the common vocabulary defined by this ontology.

We show the benefits of our ontology through an experiment in a critical infrastructure scenario, where monitoring and having a rapid response is a crucial aspect of the whole monitoring process. In this experiment, we use two devices to demonstrate the networking capabilities of our proposal: a Worldsensing commercial inclinometer node and a Raspberry Pi with an accelerometer sensor. Both pursue the same goal: to check the inclination of a railway overpass tunnel. The first device connects to our cloud server using LoRa and the other one uses WiFi as communication technology.

In summary, the main contributions of this paper are the following.

- We propose TOCA (Thing, Observation, Context, and Actuation): a semantic ontology that enables the monitoring of world entities in the IoT domain. Enabling end users to model the entire monitoring process, from sensing to actuation, including the modeling of gateways. Allowing them to access, monitor, and contextualize the information, and then perform actions as required. Our ontology is built upon existing ontologies such as SSN/SOSA, GeoSPARQL, OWL-Time, and ConAwareIoT.
- We contribute to the interoperability of things by providing a solid foundation for the development of a comprehensive and robust semantic framework, which involves enabling seamless communication and data exchange among devices and systems. Our proposal is domain independent in the IoT and, thus, everybody can benefit from adopting the proposed ontology in their systems.
- We propose an architecture for IoT, to improve the general perspective of IoT in the framework of *what, who, where, and how*: What does IoT measure? Who measures it? Where is the IoT located? How can such a system technically work?
- We provide an implementation of our approach to a scenario based on critical infrastructure monitoring, achieved by using real nodes to obtain information about the inclination of a railway tunnel, processing it, gathering the context situation, and actuating accordingly.

This paper is structured as follows: Section 2 reviews the related work. Section 3 illustrates the employed work methodology. Section 4 describes the main concepts of the ontology we propose. Section 5 matches the work done with the ontologies we build upon. Section 6 proposes an architecture for the IoT. Section 7 shows the experimentation carried out for this article. Finally, Section 8 presents our conclusions and points out future work.

2. Related work

We cover related work in the IoT monitoring domain in general, and then go into more detail to examine context awareness, semantic interoperability, and ontologies.

2.1. Monitoring

One of the inherent capabilities of IoT systems is the ability to monitor the information associated with the raw data they process. This ability is achieved by transforming raw data into relevant knowledge of the system domain. As Jennex [15] states, based on the well-known Wisdom Pyramid (Ackoff [16]), *data* is "the basic, discrete, and objective facts about something such as who, what, when, where", then data becomes *information* when "it is related to other data through a context such that it provides a useful story, as an example, the linking of who, what, when, where data to describe a specific person at a specific time". Jennex [15] also revisits the Knowledge Pyramid and adapts it to the current IoT era, where a new layer previous to *data* is added and includes *IoT and Sensors*. We agree with this proposal and assume that to enable monitoring in the IoT there is a need to implement these data and information layers.

In A. Tokognon et al. [17], the authors propose a framework for structural health monitoring using IoT devices. They define an architecture comprised of Wireless Sensor Networks (WSN) communicating via the Internet to *back-end services*. However, they focus on defining a conceptual view of the architecture from the network perspective and do not pay much attention to the *services* side. A fairly similar approach is taken by Lamonaca et al. [18], where the authors propose a monitoring framework in the same domain, centered on the WSN layers of the architecture and not on the communication from the network to cloud servers. Geetha and Gouthami [19] present a basic architecture for real-time monitoring of water quality, where they focus on data reading as such, but do not state anything specific to the server layer, beyond indicating a *data management* layer. In Malche et al. [20], the authors propose an IoT architecture for the smart cities domain. The four-layer architecture includes *Sensors*, *Gateway*, *Service*, and *Application* layers. The architecture makes sense, as it allows an understanding of the information flow and what each component contributes. However, communication between the *Sensors* and *Gateway* layers is not considered at any point, as they propose a network protocol and not a communication technology for communication.

[20] Worldsensing: https://www.worldsensing.com.

2.2. Context-awareness

IoT systems also require the ability to execute actions whenever certain conditions are met, which implies understanding the data and its structure alongside the environment to know when to react accordingly. To enable these actuation capabilities, it is necessary to obtain context information, and the primary way of achieving this is through context-awareness mechanisms. Moreover, these systems must be able to determine when there is a data anomaly to be aware of what is happening and react accordingly.

In Pillai et al. [21] the authors propose an environmental monitoring and decision support system for the IoT by using "Sensor Nodes" and "Actuation Units". However, they do not model the entities surrounding, like where are the elements located, or what are they observing. A similar situation happens in Choi et al. [22], where context-aware framework is proposed, including entities for modeling the surroundings of the events. However, they focus on a particular scenario in buildings and electrical equipment and do not provide a general approach. In Alsaig et al. [23], the authors provide a formal approach of modeling and reasoning with context knowledge, highlighting the importance of integrating context and knowledge in context-aware systems for safety-critical applications.

Regarding architectures for context-awareness in the IoT, Perera et al. [24] suggest an architecture that aims to automatically select sensor data from some user-based inputs, but they do not provide any middleware solution to maintain its context. Alagar et al. [25], establish a foundational theory for context-aware frameworks. These are defined as a broad category of mobile real-time reactive systems that continually sense their physical environment and modify their behavior in response. They also provide an analysis related to these frameworks' architecture and the components involved in a context-aware situation. Shekhovtsov et al. [26] propose a Model Centered Architecture for specifying links and interfaces in the activity recognition domain by including a modeling language that is similar to what it is done in the IoT domain, enabling to model *Things*, *Actions*, and *Properties*.

To provide an autonomous context-aware environment, Kim et al. [27] propose a system offering reduced complexity when executing if-else type rules. Including some criteria that are helpful for a few limited scenarios but they are not general enough, for instance, they do not allow combining sensors. Uddin et al. [28] introduce a conceptual framework for context-aware systems, focusing on multi-agent modeling and leveraging heterogeneous knowledge sources. The framework enables agents to extract rules from distributed ontologies, facilitating effective knowledge sharing and utilization within the system. Dobrescu et al. [29] propose a context-aware monitoring system using real-time and environmental sensors. Although, they do not clearly establish how do they facilitate the use of additional sensors beyond those they had previously considered. da Silva et al. [30] propose a three-layered architecture for the Industrial IoT: devices, gateways, and cloud services, which allows reading data from sensors, transmitting it through gateways, and representing it in the Cloud. They also map events to contextual knowledge to generate actions in the event that sensors report matching data. However, the details of the architecture are not clear, as they propose a parallel layer to the devices layer, called mobile, with similar functionalities but with a different communication technology. It is also unclear how to perform actions in this architecture, as the sensors and actuator mechanisms in IoT are not explicitly indicated. In Lunardi et al. [31], the authors propose a Human Behavior Monitoring and Support approach that provides a conceptual and semantic model for representing the user's behavior and its context in a living environment, in a similar way to that of the context-awareness paradigm by using IoT devices.

Context-awareness is a notion used also in autonomic computing systems: self-managing and self-adaptive systems that attempt to reduce human intervention in the operation and maintenance of complex systems. These systems may monitor and analyze their own behavior, make judgments, and take actions in order to improve their performance, attain specific goals, or adjust to changing conditions [32].

2.3. Semantic interoperability and ontologies

As stated in the Wisdom Pyramid [16], *knowledge* means "information that has been culturally understood such that it explains the how and the why about something or provides insight and understanding into something". One way of approaching is by means of semantics, as stated by Noura et al. [11] who propose a taxonomy to classify the levels of interoperability as *device*, *syntactic*, *network*, *semantic*, and *platform*.

The Open Geospatial Consortium (OGC) published a series of standards known as "The Sensor Web Enablement" that aim to ensure interoperability between sensor and actuator systems within the IoT domain. O&M[21] and SensorML[22] are two of the most relevant standards in this framework. O&M defines standard XML Schema models for observations and features related to sensor measurements, while SensorML provides a semantic and robust way to define the characteristics and capabilities of sensors, actuators, and computational mechanisms. Overall, these standards have played a crucial role in addressing interoperability challenges in the IoT domain. The Semantic Sensor Network Incubator Group (SSN-XG) developed the Sensor and Sensor Network (SSN) [33] ontology to enable applications interoperability by semantically describing sensors and network resources, where they combine sensor-focused (SensorML), observation-focused (O&M), and system-focused views to provide a domain-independent and end-to-end model for sensing applications. Shortly thereafter, the World Wide Web Consortium (W3C) recommended using an updated version of the SSN ontology, known as SOSA (Sensor, Observation, Sample, and Actuator) [34], for cases that require Semantic Web and Linked Data technologies, since SOSA is a lightweight and self-contained core ontology of the SSN.

[21] Observations and Measurements (O&M): https://www.ogc.org/standards/om.

[22] Sensor Model Language (SensorML): https://www.ogc.org/standards/sensorml.

In accordance with Rhayem et al. [35], with the emergence of the IoT paradigm, a vast quantity of real-world objects are currently connected to the Internet and exchanging data with each other. In their review, they provide a classification schema and a review of the literature for the Semantic Web. Mountrouidou et al. [36] propose a taxonomy for the IoT where they incorporate features of IoT devices as well as their purpose, and also sensor and actuator concepts. However, they focus on the "mobility" of the device and the communication channels, leaving aside very important entities such as the object or element of the real world on which they are observing or actuating. CAMeOnto [37] proposes an ontology that contains the *user*, *activity*, *time*, *device*, *services*, and *location* entities. Yet, the relationship between what is observed and what is acted upon is unclear, as they do not account for the sensorization component. In MSSN-Onto [38], an ontology is proposed to represent sensors, events, and the surrounding elements. They employ SOSA/SSN as their foundational ontology but stop at its *observation* perspective, without taking into account the *actuation* perspective. There is the NGSI-LD [39] ontology, an ETSI standard that allows sharing of schema information with context information but does not support actuation environments. Additionally, Kiljander et al. [40] proposes the interoperability of semantic information brokers using the IoT-A[23] project, which supports the sensor and actuation aspects but does not consider context management to model context-awareness rules.

2.4. Related work summary

From the previous analysis, we can conclude that monitoring in the IoT domain still requires further research. Proposals in the Interoperability of Things do not take into account all identified needs. A similar situation happens when looking for ontologies covering the main aspects required, i.e., sensors, actuators, and context-awareness, since none of the existing proposals covers jointly the three topics we address in this paper. The situation is even worse when looking for proposals dealing with gateways.

3. Research methodology

We apply the *Knowledge-Engineering Methodology* [12] to design and implement our proposal. The methodology has inspired us to create correct semantics for the IoT domain. Semantics can be developed in a variety of ways, and domain modeling is affected by several factors, including the overall goal of the system. Moreover, the modeling of an ontology is an iterative process and there are some steps to achieve a successful ontology definition. Below is a summary of these steps and how we have approached them:

1. **Determine the domain and scope to be covered**
   The domain and scope of the framework have been identified in this research, particularly in Section 1.
2. **Sketch a list of competency questions that should be answered**
   We provide in Section 3.1 a list of the competency questions that are answered by our ontology.
3. **Consider reuse of existing ontologies**
   A review of similar existing ontologies and related research is given in Section 2. In addition, a detailed explanation of how we extend the existing ontologies is given in Section 5.
4. **Enumerate important terms in the ontology**
   The important terms and entities have been thought of while the ontology has been developed, stated in Section 4.
5. **Develop the ontology**
   Our ontology has been developed in accordance with what is written in this paper. In particular, Section 4 provides a conceptual overview. Section 6 proposes an architecture for IoT, and Section 7 performs an experimentation, both considering the ontology.
6. **Validate the list of competency questions**
   Our proposed competency questions are formulated in Section 3.1 and validated in Sections 4, 6, and 7.

3.1. Competency questions

Determining the scope is a crucial step in the development of an ontology, and an effective method to achieve it is to identify a set of "competency questions", designed to establish the baseline knowledge required for the ontology to be useful. By answering these questions, the knowledge engineer can gain a clear understanding of the boundaries and content of the ontology. If the ontology can provide accurate and complete answers to each competency question, it is a sign that the ontology contains sufficient information to be effective in the intended application. Following are the competency questions that we have considered:

- **Q1**: What kinds of devices can be connected to an IoT network?
- **Q2**: If the user needs to know which gateways are being used by which IoT devices for each measurement received in the system, how can this be accomplished?
- **Q3**: How can devices, sensors, and actuators be grouped?
- **Q4**: How are the actuations indicated? Who performs them?
- **Q5**: How can one keep track of the actuations performed?

[23] IoT-A: https://www.iot-a.eu.

- **Q6**: How do IoT devices enable context awareness?
- **Q7**: How can the entities be accessed and managed?
- **Q8**: How can data from IoT devices be collected, stored, and analyzed to gain valuable insights?

When building our *competency questions*, we have considered the needs of research both in an academic environment and in an industrial environment since they are included in a real use case for Worldsensing.

4. Our proposal - The TOCA ontology

To facilitate the widespread adoption of the Internet of Things (IoT), a context-aware monitoring framework is an essential requirement. To achieve this adoption, a clear understanding of the information flow between end devices, namely Things, and cloud servers is necessary. Hence, the ontology we propose emphasizes the data communication and semantic aspects of the IoT. Our goal is to introduce a novel approach that enhances IoT semantic interoperability with a specific focus on device management and entity monitoring. Our ontology also includes the modeling of gateways, which are crucial components of the IoT technologies to enable cloud communication with end devices.

In this section we provide an overview of the concepts in our ontology and describe how are they related to each other. Then, we explain these concepts in detail and the means to access them via resource URIs.

4.1. The TOCA ontology

The TOCA (Thing, Observation, Context, and Actuation) ontology is aimed at enhancing interoperability for the monitoring of physical infrastructures by using IoT devices and by providing mechanisms for enhanced interoperability between IoT entities. Our proposal is based on three main points: **(1) Sensing and observation**: the knowledge of the entities that obtain measurements of the environment they are monitoring, like *Sensor*, *FeatureOfInterest* or *Observations*; **(2) Contextualization and actuation**: the entities that perform actions based on previous knowledge, obtained in (1), by using *Actuators*, *ContextAwareRules* and the required steps to evaluate the rules that check the data provided by the sensors and then, executing actions when needed; **(3) Information management**: the metadata of the elements that perform the sensor and actuator parts, like *Things* or the data on network communication provided by *Gateways*, including information about location or placement in the environment using *Groups*.

TOCA considers the dynamic nature of IoT systems, embracing flexibility and adaptability as core principles, where IoT systems are characterized by their ability to adapt and change over time, enabling the full potential of these interconnected systems. As technology continues to evolve, we may currently discuss LoRa, but in the future, it could be cellular networks or even another type altogether. Nevertheless, our proposed ontology is designed to effectively model these relationships, regardless of the evolving technological aspects or changes in sensing and actuating functionalities. TOCA provides a way to represent entities for the evolving nature of IoT systems, enabling the seamless integration of new devices joining the network and devices that may leave. When a device exits the system, it stops reporting data to the system. Similarly, when a new device joins, a new instance is created to incorporate its measurements. Furthermore, the ontology takes into account the diverse capabilities and configurations of IoT devices, providing a comprehensive and scalable model to accommodate the entire range of devices in the ecosystem.

Fig. 1 illustrates our ontology specified by means of a UML class diagram. This diagram states the class names and relationships between classes and includes namespaces to specify the origin of the concepts in the TOCA ontology. For example, the notation *(sosa:Sensor)* or *(sosa)* below *Sensor* indicates that the TOCA concept *Sensor* is extended from the one in the SOSA ontology with the same name. The diagram highlights the elements that we have introduced or updated in our proposal, as related to previous ontologies. For a comprehensive explanation of all concepts in the TOCA ontology, please refer to Section 4.2.

TOCA also includes three textual integrity constraints that establish conditions that any instance of the ontology must meet: (C1) A Location must have at least one Feature or one Geometry; (C2) The Time of Observation must be the same as or before the Time of Actuation; (C3) A Sensor can only be hosted by one Thing or Gateway at the same time.

4.2. Description of the concepts in TOCA

The TOCA ontology extends our previous *ConAwarIoT* ontology [41], which was proposed to deal with context awareness in the sensing and actuation domains by using IoT devices. *ConAwarIoT* incorporates concepts for context-aware actions, but we did not indicate clearly enough the relationships of the context-aware entities with the entities performing the actions and, in general, for keeping track of the monitoring status. In a similar way, *ConAwarIoT* draws from the *IoTMA* [42] ontology, which is oriented towards the understanding of sensor-actuation contexts and the concepts of general monitoring and actuation.

In this section, we explain the new concepts introduced in TOCA to holistically manage interoperability, context awareness, and the state of the communication; as well as those we have modified or updated from the ontologies we extend. We refer the reader to the original ontology description for the rest of the concepts.

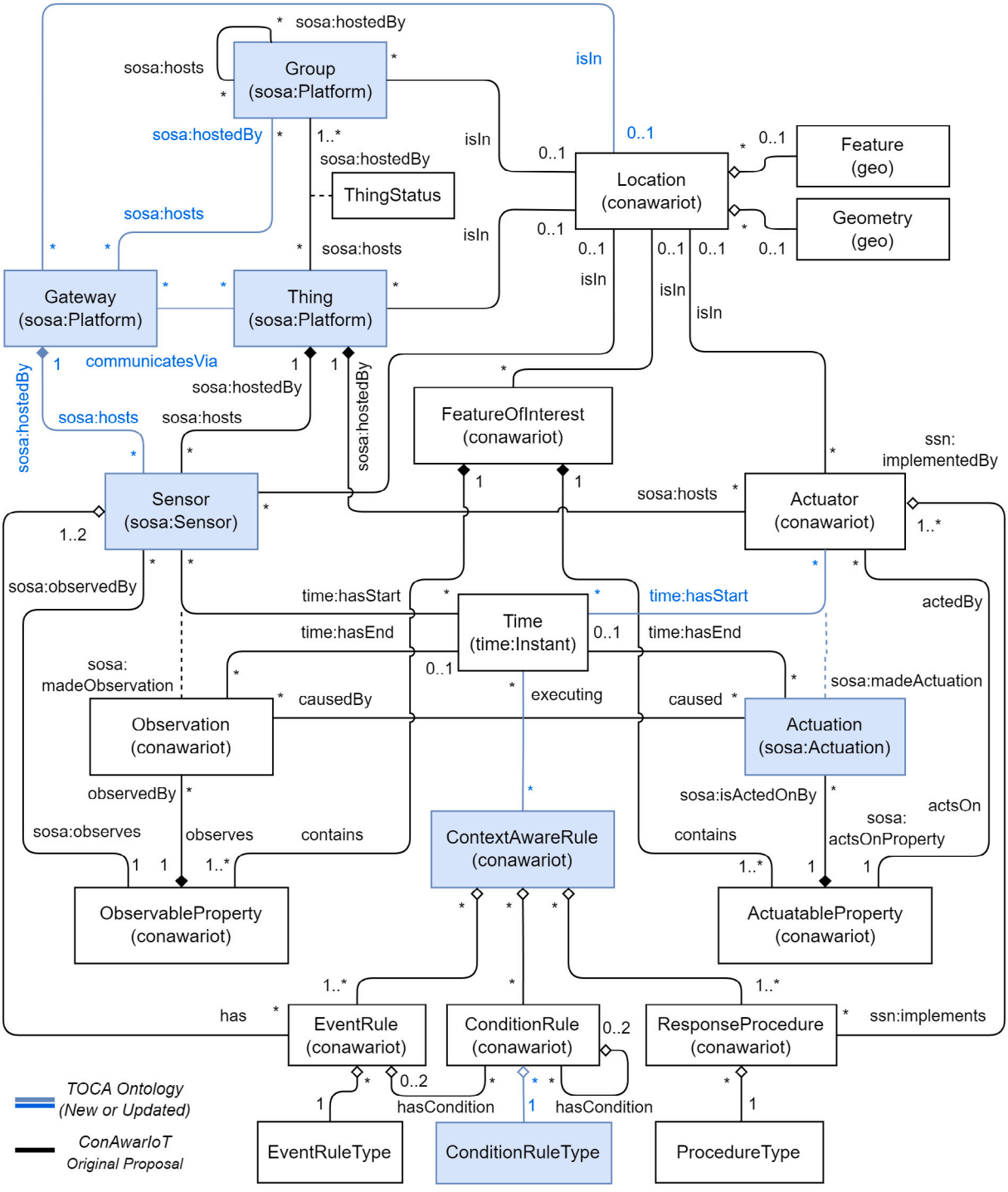


Fig. 1. Overview of the TOCA ontology.

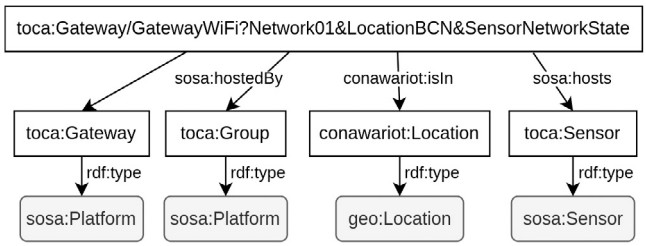


Fig. 2. Example of a Gateway description using RDF graphs.

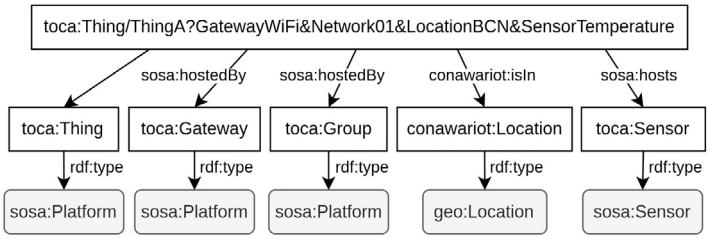


Fig. 3. Example of a Thing description using RDF graphs.

4.2.1. Gateway

A *Gateway* is a network element that communicates *Things* with entities in the Fog or Cloud, following the Cloud Continuum. This entity can report information from its *Sensor*s, as well as redirect the information sent to it by the *Thing*s. An example of a *Gateway* can be a WiFi router, but also a LoRa or an NB-IoT gateway. As shown in Fig. 2, an example of a Gateway is that of one with the name "GatewayWiFi", the group that is hosting it "Network01", where it is located "LocationBCN" and the sensors it does handle, in this case, one, the "SensorNetworkState".

This entity along with *Group*, *Thing*, *Sensor*, and *Actuator*, provides a response to *Competency Question 1* and also addresses *Competency Question 2*.

4.2.2. Thing

We stated in *ConAwarIoT* that a *Thing* "is an element that reports information employing sensors and can have actuation capabilities.". Even though this definition is still valid, we propose here a new way to modeling *Thing*s. The idea now is that a *Thing* no longer knows its *Observations* through a direct relation since the *Sensors* are connected either way to a *Thing*. Therefore, *Observations* are only communicated in that way, via *Things*. We also establish the relationship with the new entity *Gateways*. A *Thing* can communicate, or it is connected, or it depends on a *Group* directly, but it can also depend on a *Gateway*. This brings flexibility to the modeling because now a *Thing* is then considered network-independent.

In Fig. 3, we observe a composition of a *Thing* named "ThingA", which is hosted by the "GatewayWiFi", meaning that the system is aware that information flows through that *Gateway*, although the number of *Gateway*s is not limited. There is also a *Group* hosting this *Thing*, the "Network01"; the *Thing isIn* "LocationBCN" and hosts a *Sensor*, named "SensorTemperature".

This entity, together with *Group*, *Gateway*, *Sensor*, and *Actuator* answers the *Competency Question 1*.

4.2.3. Group

This entity is an enhancement from the *ConAwarIoT* ontology under the entity name *Platform*. We have now renamed it to *Group*, so that it can be better understood as an entity that is able to represent and group any subset of elements, *Things* or *Gateways*, employed to observe and actuate on world entities. A *Group* is responsible for hosting (*sosa:hosts*) *Things* and *Gateways*, and also applying the reflexive conditions of the *sosa:Platform*, where a *Group* can host another *Group*, thus enabling multiple *Group* level grouping or more structured management. For instance, a *Group* can be either a building, a machine, a group of machines, and as well a group of machines or even a group that handles buildings and machines because it is a construction site, and so on. The entity is related to *ThingStatus* to allow handling the status of a *Thing* for each *Group*. A *Group* is identified by a unique name and contains its location information to allow them to be placed, as depicted in Fig. 4.

This entity, together with *Gateway*, *Thing*, *Sensor*, and *Actuator*, answers the *Competency Question 1*. It also covers part of the *Competency Question 3*.

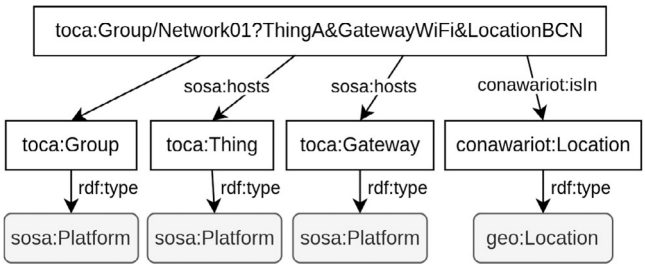


Fig. 4. Example of a Group description using RDF graphs.

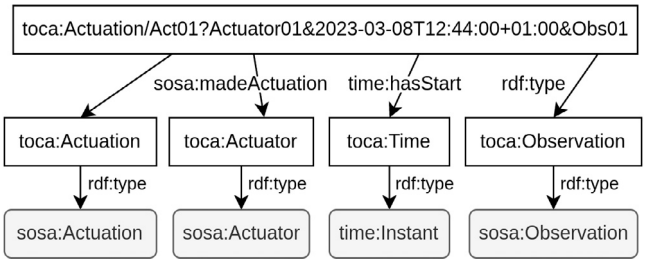


Fig. 5. Example of an Actuation description using RDF graphs.

4.2.4. Sensor

We update here the definition provided in *ConAwarIoT*: "*Sensors* are elements capable of making raw measurements (Observations) of a given property (ObservableProperty).". This definition is still valid, but we want to remark that sensors, in TOCA, can be stated as metadata features of the entities that *sosa:hosts* in *Sensor*. For example, a *Sensor* can be the state of the network connection for a *Thing*; or the signal strength with which a *Thing* transmits to a *Gateway*; or even a *Gateway* sensor, such as the temperature.

This entity, together with the *Group*, *Gateway*, *Thing* and *Actuator* answers the *Competency Question 1*.

4.2.5. Actuator

*Actuator*s are the elements that execute *Actuation*s to change the state of an element of the world in a particular moment of *Time*. These *Actuation*s are executed to affect an *ActuatableProperty*. The definition provided in *ConAwarIoT* is still valid but we propose here a change related to the *Actuation*: similarly to the *Sensor-Observation*, an *Actuator* is now only able to execute one *Actuation*.

This entity, together with *Group*, *Gateway*, *Thing*, and *Sensor*, answers the *Competency Question 1*.

4.2.6. Actuation

We redefine the specification of this concept in *ConAwarIoT* since now an *Actuation* is an execution of an *Actuator*'s task, at an instant of *Time*, and it modifies an *ActuatableProperty*. This *Actuator* receives the action requests via the *ResponseProcedures*. An example of an *Actuation* is, adjusting the temperature of a car or sending an email to warn someone. Fig. 5 illustrates an example, where an *Actuation* is created, from the *Actuator01*, and it includes the *Time* that has started, as well as the *Observation* that provoked it.

This entity, answers the *Competency Question 4* and the *Competency Question 5*.

4.2.7. ContextAwareRule

We slightly modify the *ConAwarIoT* definition of this concept. Now, *ContextAwareRule* entity is used to define a set of steps to check certain events, conditions, and responses for further action. Therefore, the relationship it has with the *Time* entity is to perform these checks and state fields as the *executing* state, but not to automatically generate an *Actuation*, which is now handled by the *ResponseProcedure*.

This entity, together with *EventRule*, *ConditionRule*, and *ResponseProcedure*, answers the *Competency Question 6*.

Table 1
Summary of the main URIs in TOCA.

Class	URI patterns
Group	`TOCA:Group/{GroupName}?{LocationName}&{Groups}&{Things}&{Gateways}`
Thing	`TOCA:Thing/{ThingName}?{LocationName}&{Sensors}&{Actuators}&{Groups}&{Gateways}`
Gateway	`TOCA:Gateway/{GatewayName}?&{LocationName}&{Sensors}&{Things}&{Groups}`
Sensor	`TOCA:Sensor/{SensorName}?{ThingName|GatewayName}&{ObservablePropertyName}&{LocationName}&{Observations}`
Actuator	`TOCA:Actuator/{ActuatorName}?{ThingName}&{ActuatablePropertyName}&{LocationName}&{Actuations}`
FeatureOfInterest	`TOCA:FeatureOfInterest/{FeatureOfInterestName}?{ObservablePropertyNames}&{ActuatablePropertyNames}&{LocationName}`
ObservableProperty	`TOCA:ObservableProperty/{ObservablePropertyName}?{ObservationValueType}&{FeatureOfInterestName}&{Sensors}`
Observation	`TOCA:Observation/{ObservationID}?{SensorName}&{TimeStart}&{ActuationIDs}`
ActuatableProperty	`TOCA:ActuatableProperty/{ActuatablePropertyName}?&{FeatureOfInterestName}&{Actuators}`
Actuation	`TOCA:Actuation/{ActuationID}?{ActuatorName}&{TimeStart}&{ObservationIDs}`
ContextAwareRule	`TOCA:ContextAwareRule/{ContextAwareRuleName}?{EventRulesName}&{ConditionRulesName}&{ResponseProceduresName}&{Executing}`
EventRule	`TOCA:EventRule/{EventRuleName}?{ContextAwareRuleName}&{EventRuleTypeName}&{Sensor1Name}&{Sensor2Name}&{ValueBoolean|ValueString|ValueInteger|ValueFloat}`
ConditionRule	`TOCA:ConditionRule/{ConditionRuleName}?{ContextAwareRuleName}&{EventRule1Name}&{EventRule2Name}&{ConditionRule1Name}&{ConditionRule2Name}&{ConditionComparationType}`
ResponseProcedure	`TOCA:ResponseProcedure/{ResponseProcedureName}?{ContextAwareRuleName}&{ProcedureTypeName}&{ActuatorNames}`
Location	`TOCA:Location/{LocationName}?{Feature|Geometry}`
Time	`TOCA:Time/{Date}T{Time}+{TimeZone}`

4.3. Resource URIs

We provide URIs[24] in Fig. 1 to increase the simplicity and manageability of systems that are defined according to our ontology. The design of these URIs is represented in Table 1, where the first column represents the entity's class name in our ontology. The second column represents the proposed URI pattern, defined as: `{BASE_URI}:{CLASS_NAME}/{CLASS_ID}?{CLASS_PROPERTY_1}&{CLASS_PROPERTY_N}`. Where the `BASE_URI` denotes the URL entry point, located in a test domain. The `CLASS_NAME` indicates the name of the entity, following the `CLASS_ID`, which is the identifier for each instantiation of an entity. In addition to the `CLASS_PROPERTY_N`, which contains the instantiated entity's N properties.

With the contents of this table, we comply with the *Competency Question 7*.

5. Ontologies extended by our proposal

Our ontology is proposed as an extension of various existing ontologies and query languages, that cover different but complementary aspects that must be considered to provide a solution to the problem at hand. We have used four of the existing ontologies for the following purposes:

- ConAwarIoT: ontology used as a basis for our ontology.
- SSN/SOSA: ontology used to gather measurements from sensors and act using actuators.
- GeoSPARQL: standardized query language used to describe geographical locations.
- OWL-Time: ontology used to describe temporal properties.

[24] Uniform Resource Identifiers (URI): Unique sequences of characters that identify a logical or physical resource.

5.1. ConAwarIoT - Context-Aware responsive IoT ontology

TOCA extends functionalities from *ConAwarIoT* [41] by proposing *sensors* and *actuators* entities. Additionally, it allows modeling context-related actions in the IoT through entities that enable the composition of rules by providing events, conditions, and the actuations to execute when specific conditions are met.

ConAwarIoT ontology is already an extension of the Semantic Sensor Network ontology (SSN) and the Sensor-Observation-Sample-Actuator (SOSA) ontology, in addition to the OGC GeoSPARQL and OWL-Time. *ConAwarIoT* incorporates terms such as *Site* to group *Thing*s; *Thing*s are either *Sensor*s or *Actuator*s; The properties to observe and act on *ObservableProperty* and *ActuatableProperty*, as well as their corresponding *Observation*s and *Actuation*s. Related to context-awareness, it *ConAwarIoT* incorporates terms such as *EventRule* to model the events and measurements to check; *ContitionRules* which are the conditions to occur; and the *ResponseProcedures*, in case something happens, then executes a procedure.

5.1.1. About actuations and ContextAwareRules

In *ConAwarIoT*, an *Actuation* is given by an instant of time in which a result is obtained from an execution of a *ContextAwareRule*. However, in TOCA, *Actuations* are generated from *ResponseProcedures* which occurs once the events and conditions established in the *ContextAwareRules* have been checked. In addition, for simplicity, an *Actuation* can only be generated by an *Actuator* at a given time instant, and this *Actuation* will change the state of one *ActuatableProperty* entity.

5.1.2. Adding gateways

In *TOCA* we have incorporated the modeling of this entity: the *Gateways*, that is, the elements responsible for transmitting the information in many of the technologies used in the IoT paradigm. Therefore, users of our proposal can model context-aware actions considering those entities as well.

5.1.3. Platform to group

We have changed the name of this entity to bring more clarity to its purpose, as in the end, the entity is meant to be used to group other entities as *Things* and now *Gateways*.

5.1.4. Observations reported to sensors and not sensors and things

We propose to remove the relationship between *Observations* and *Things*, as we understand that it is redundant since the *Sensor* itself (to which a *Thing* may belong) already has this knowledge.

5.1.5. Remove ThingType

For a clearer and more understandable view of our proposal, we have decided to keep this entity as an attribute, as stated in the previous sections. Hence, it is no longer illustrated in the conceptual diagram.

5.2. SSN/SOSA ontology

SSN/SOSA [33,34] is the ontology that is closer to the concepts modeled here as it provides a lightweight core for defining common classes and properties of data managed in the IoT domain. It supports both sensing and actuation capabilities, which still provide a perfect fit for modeling interoperability in the IoT. The SSN/SOSA classes that have been (re)used in our solution, apart from those already stated in *ConAwarIoT*, are the *sosa:Platform*, used to describe *Gateways*; and the *sosa:Platform*, used to describe *Groups*.

5.3. OGC GeoSPARQL

We also use the GeoSPARQL [43] ontology to define the spatial attributes of the concepts in our proposal. GeoSPARQL allows defining both the detailed representation of the element at the coordinate level (geo:Geometry) about points, lines, or polygons; and a description of the representation, e.g., “Barcelona”.

5.4. OWL-time

OWL-Time [44] is used in TOCA to describe temporal concepts and properties. *Time* is used to define when there is an *Observation* or an *Actuation* (*time:hasStart*). If necessary, the end time (*time:hasEnd*) can also be indicated. In addition, through the use of *time:Instant* it becomes possible to specify the precise instant when certain *Observation* or *Actuation* events occur.

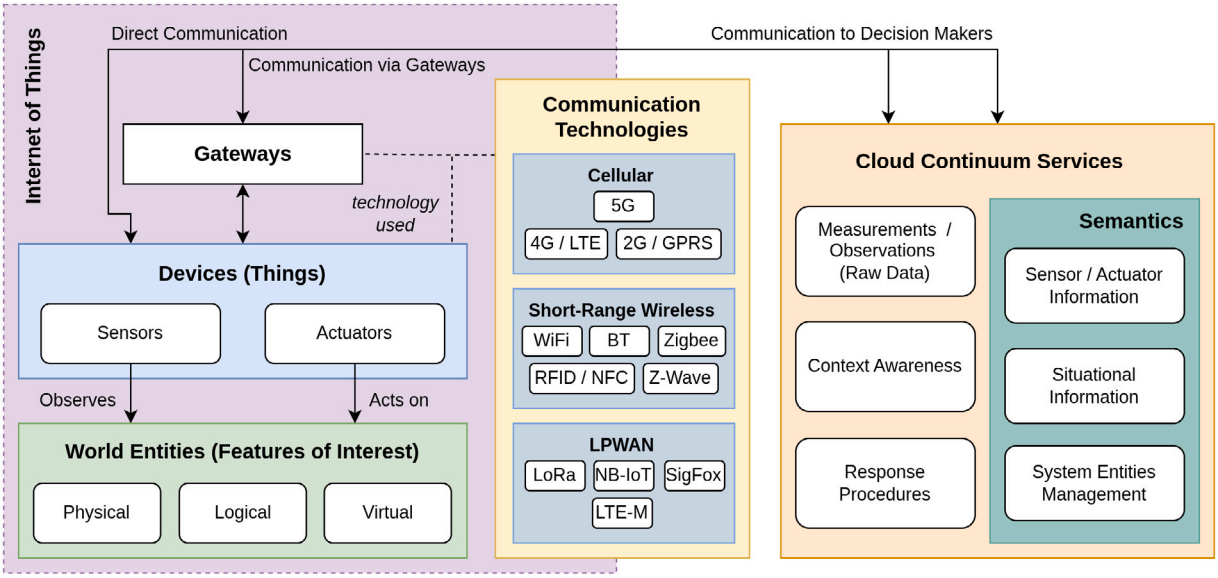


Fig. 6. An IoT-based architecture for entities monitoring.

6. An IoT-based architecture for awareness monitoring

IoT revolves around the connection and communication of data among physical devices and systems, enabling real-time data collection, analysis, and the planning of response procedures. On the other hand, for individuals new to the field, understanding the broader scope of IoT systems can be challenging due to their inherent complexity. In this section, we propose an architecture for IoT systems, examining the key components and how they work together to enable IoT applications. We focus on a conceptual architecture that handles concepts from both the telecommunications and the computer science domains.

Fig. 6 illustrates our proposal. The starting point is the *World Entities*, namely Things: descriptions of both physical and logical elements. Then we find the *Internet of Things* concept as such, i.e., the set of devices that allows us to obtain information from *Things* and act on them. At this point, the data measurements are already in these devices and are ready to be communicated to the servers. Before that, communication has to be established, probably using *Gateways* or other *Communication Technologies* in the same way. Once the communication is established, *Thing*'s data is sent to the *Cloud Services* where the data is manipulated to extract knowledge and also contextualize and plan responses. All processes are performed using data structures with standardized or semantic representations. Once the system is up and running, it can contact the IoT elements or gateways again, for example, to execute certain actions.

6.1. Internet of things

We understand the *Internet of Things* as the set of all entities in the physical, logical, and virtual worlds; the devices used to obtain information through sensors and act on the previously described entities; the devices used to transmit the information to the Internet; and, to a certain extent, the communication technologies that enable efficient exchange of this information.

6.2. World entities

The target entities to be monitored in the entire IoT domain, where they are the *features of interest* that the environment implementers are interested to monitor. We divide them into the three categories:

- Physical Entities: They are real-world physical objects or elements being sensed and actuated upon by IoT devices. Examples of physical entities include temperature sensors, smart thermostats, smart home appliances, and industrial equipment, as well as the elements being monitored.
- Logical Entities: They are the software abstractions of the physical entities that IoT applications create to enable the processing and analysis of sensor data. Examples of logical entities include data models, ontologies, and metadata such as the network communication status.
- Virtual Entities: They are the digital representations of the physical entities created along the cloud–edge continuum. Examples of virtual entities include digital twins, which are virtual replicas of physical devices or systems, and the simulations that model their behavior.

6.3. Devices (Things)

The physical objects or systems, also known as *Devices* or *Things* that are equipped with sensors, actuators and communication technologies that enable them to collect and exchange data over the Internet. These devices range from simple sensors that measure temperature or humidity to complex machines or vehicles with a wide range of sensors and actuators that enable them to perform a variety of functions.

IoT systems use several types of reporting technologies, including event-driven, synchronous, asynchronous, publish–subscribe, and peer-to-peer communication. For example, event-driven communication occurs when devices send data only when specific events occur, like a temperature sensor that sends data when the temperature exceeds a particular threshold. On the other hand, asynchronous communication is when data is sent and received independently of any clock signal or timing mechanism like a smart thermostat sending data to a central hub whenever the temperature changes. Pub/Sub communication occurs when data is published by a publisher device to a message broker and then subscribed to by one or more subscriber devices, like an industrial control system. Finally, peer-to-peer communication occurs when devices communicate directly with each other without a central server, such as a fleet of industrial robots in a factory, using machine-to-machine communication to coordinate their movements to avoid collisions.

6.4. Gateways

A gateway is a physical element that acts similarly to a router, gathering data from incoming wireless transmissions and submitting it to the server. Depending on the communication protocol used, using at least one gateway can be a requirement for the system; more than one, or maybe optional. Gateway typologies in the most common communication technologies can be classified as follows.

- WiFi (Short-Range): In this technology, the standard form of communication is facilitated through the use of an Internet router. The IoT device, for instance, a Raspberry Pi, knows the communication credentials of that gateway. Then, to communicate, it has to establish a connection and then transmit. If the device moves or loses connection to this router, and it is no longer available, the device will need to establish a new connection to another router, performing the entire authentication process again.
- Cellular: Gateways in this type of communication are transparent to the user at the management level. The IoT device, on its *SIM card*, has credentials to communicate with a particular operator's infrastructure or antenna network (gateways), and not credentials for a particular antenna. The authentication is performed once, and if authorized, the device can reuse that authentication for other antennas from the same operator, establishing a handover process. With this technology, the device is only explicitly connected to one antenna, at any given time.
- LoRa: Low-Power Wide-Area Network (LPWAN), where the gateways in this form of communication are similar to those of cellular networks. The most notable difference is that the devices do not establish a connection with a specific gateway to communicate. Instead, they emit a radio message and the gateways that are within range listen to it. Previously, the device had to be configured with the credentials of that gateway network as well as with the application key to where it is providing data.

6.5. Communication technologies

Communication technologies play a crucial role in the IoT because they enable devices to exchange with each other and with cloud-based applications. There is a great diversity of ways to communicate in IoT, some focused on efficiency and low power consumption, some on latency, and others on generalization, to be used in different environments. In our architecture, we contemplate three categorizations:

- Cellular: The most popular, as they are used for the most common communications, such as phone calls. These have the advantage of already having an infrastructure in place, as they are the oldest and have been improved on a regular basis to adapt to new times. On the other hand, can have limitations in terms of power consumption, cost, and coverage in some areas, particularly in remote or rural areas. Examples of these networks include 5G, 4G/LTE, and 2G/GPRS.
- Short-Range: They have a relatively limited range of coverage, typically from centimeters to less than 100 m. These networks are typically used in applications where devices need to communicate over short distances and require low latency, high data rates, and high reliability. Examples of these networks include WiFi, Bluetooth, RFID/NFC, Zigbee, and Z-Wave.
- LPWAN: Low-Power Wide Area-Networks are designed for long-range communication and can cover distances of up to several kilometers. They are specifically designed for IoT applications that require low power consumption, long battery life, and low data rates. Examples of these networks include LoRa, NB-IoT, Sigfox, and LTE-M.

Fig. 7. Monitoring a rail-way tunnel.

6.6. Cloud continuum services

In our architecture, we provide a comprehensive approach that encompasses services, components, ontologies, semantics, and infrastructure management tools for monitoring IoT systems. To ensure scalability and flexibility, we embrace the Cloud Continuum concept, which encompasses all computing tiers from the Cloud to the Thing level and also recognizes the importance of intermediate tiers, such as the Edge and the Fog, which play a crucial role in enabling efficient processing and decision-making closer to the sources.

The Thing tier, central to our proposed IoT-based architecture, represents the core devices and entities within the system. The Cloud tier, on the other hand, refers to the more traditional cloud-based infrastructure for centralized processing and storage. Our architecture, however, allows implementers the freedom to introduce a Fog tier between the Edge and the Cloud, which can provide functionalities similar to those of the Cloud but with the advantage of reduced latency and enhanced local processing capabilities.

In our proposal, the services listed below can be at any point in the cloud-to-thing continuum:

- Measurements and Observations: This component encompasses all the services required to receive, process, and store the raw data sent by IoT devices.
- Context Awareness: Once we have the data in the system, this module has the contextual knowledge of the entities to monitor, where the values received into the system are checked, to react if necessary.
- Response Procedures: Once an action is detected, it has to be reacted to and the action has to be executed. This is the purpose of this component, allowing the definition of the actions to perform.

For these services to work there must be a semantical agreement that allows correct identification of the manipulated entities, to do so, we define the following services, making use of semantics and ontologies:

- Sensor and Actuator Information: This component enhances interoperability and represents the semantics necessary to understand the measurements received from sensors in a raw form.
- Situational Information: This component represents the semantics needed to understand the context of the measurements received along with the system state empowering interoperability.
- System Entities Management: All auxiliary but necessary components are contemplated to guarantee the correct maintenance of the system, such as the location and the elements that are observed.

7. Experimentation

To show the feasibility of our approach, we have developed a context-aware solution for the monitoring of entities on the IoT. Moreover, with this, we are also able to show in practice the advantages that ontologies offer when applied to the Cloud Continuum paradigm. For this purpose, we have established the foundation for our experimentation in IoT device interoperability in the critical infrastructure domain. By using the concepts defined in our ontology, we facilitate the exchange of data between sensing and actuation devices.

Our focus lies on the domain of critical infrastructures. These infrastructures are of great strategic importance, as they provide essential services that are indispensable for civil infrastructure projects such as dams, mines, railway tunnels, and construction sites. To minimize the possibility of failure, a continuous monitoring of these is imperative. Our practical use case is illustrated in Fig. 7. We aim to communicate data measurements by using different types of devices and networks, performed using the same *Cloud Services* and taking advantage of the ontology that we propose. To do so, we merge the concepts of the IoT architecture presented in Section 6 with the implementation of the ontology proposed in Section 4.1. We answer the *Competency Question 8* with this experimentation.

7.1. Case study

Our case study is the monitoring of critical infrastructures as illustrated in Fig. 8. Where we need to know the status of both the monitored elements and the ones performing the monitoring. We use two devices to monitor: a commercial Worldsensing IoT

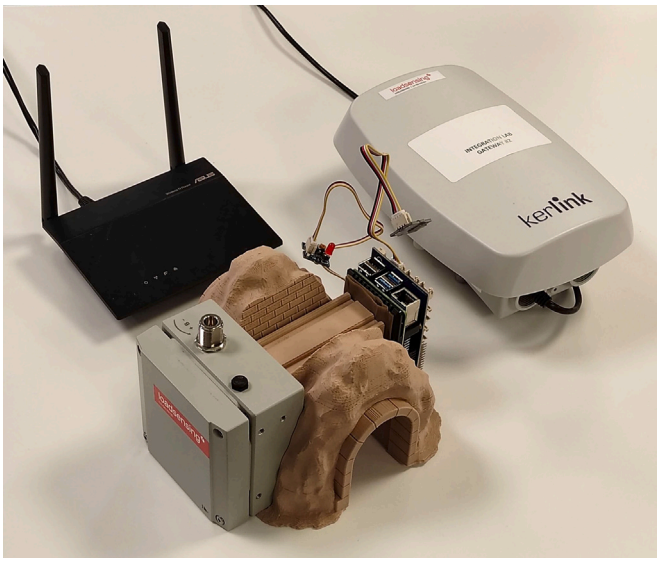


Fig. 8. Our Raspberry Pi, a Worldsensing IoT node, and two gateways.

Tiltmeter[25] node that measures the inclination and a Raspberry Pi,[26] with a Grove Pi Shield[27] with a tilt sensor considered as a critical sensor.[28]

To recreate the test environment, we have a 3D-printed replica of a tunnel next to a train track on top of a mountain. On each side of the replica there is one of the two IoT devices mentioned above. The Worldsensing's IoT Tiltmeter node (leftmost gray element) communicates information via the Kerlink LoRa gateway (rightmost element), while the Raspberry Pi uses the ASUS WiFi router, to communicate data measurements with the Cloud server, using our specifications mentioned in the ontology. By incorporating various devices and technologies into our platform, we can ensure efficient and effective data acquisition from multiple sources.

The inclinometer manufactured by Worldsensing takes measurements at regular intervals and transmits them to the Cloud via LoRa. The gateway is responsible for collecting the information and relaying it to the cloud server. In addition, the Raspberry Pi also takes measurements and communicates them over WiFi, cloud communication, enabling effortless data transmission to the Cloud.

7.2. Experimentation architecture and infrastructure

Our experimentation incorporates several software components that are encapsulated in functional modules. As we make use of a microservice architecture, each module has its own process and communicates with others using lightweight mechanisms, such as HTTP requests. This approach facilitates the scalability, interoperability, modularity, portability, and extensibility of the project across different computing environments. Allowing independent deployment and scaling of smaller system components.

The Cloud infrastructure comprises a *GCP E2-small* instance running Debian 11 Linux with 1 vCore and 2 GB of RAM. We have deployed our custom server code, which adheres to the proposed TOCA ontology, in this instance. A detailed explanation of our code and software block description is available in Section 7.3. Under normal operating conditions, the server load on the *GCP* instance ranges from 10% to 25% CPU usage while updating data measurements, and the RAM usage is around 1 GB.

Fig. 9 illustrates the mapping between our proposed IoT architecture and the outcome of our experimentation. The figure displays the entities that, in our view, constitute the definition of IoT, along with the corresponding elements to be monitored, as well as the cloud services system's software components.

When a user, whether human or device, initiates communication with our system, the first point of contact is the API Gateway, which is powered by *Kong*.[29] It receives the request and directs it to the appropriate service. For example, if the request is to update the front-end, then it will communicate with the *NGINX + React* service. If the request involves querying or updating the status of an element in the database, it will communicate with the relevant back-end service, developed using *Python FastAPI*.[30] The back-end consists of multiple endpoints that make use of the ontology we have designed, corresponding to the "Semantics" box in Fig. 9.

[25] Worldsensing Tilt90-x: https://www.worldsensing.com/product/tilt90-x-2/.
[26] Raspberry Pi: Small computers with limited computing capabilities. https://www.raspberrypi.org.
[27] Grove Pi: https://www.seeedstudio.com/GrovePi.html.
[28] 6-Axis Accelerometer and Gyro.: https://www.seeedstudio.com/Grove-6-Axis-Accelerometer-Gyroscope.html.
[29] https://konghq.com/products/api-gateway-platform.
[30] https://fastapi.tiangolo.com.

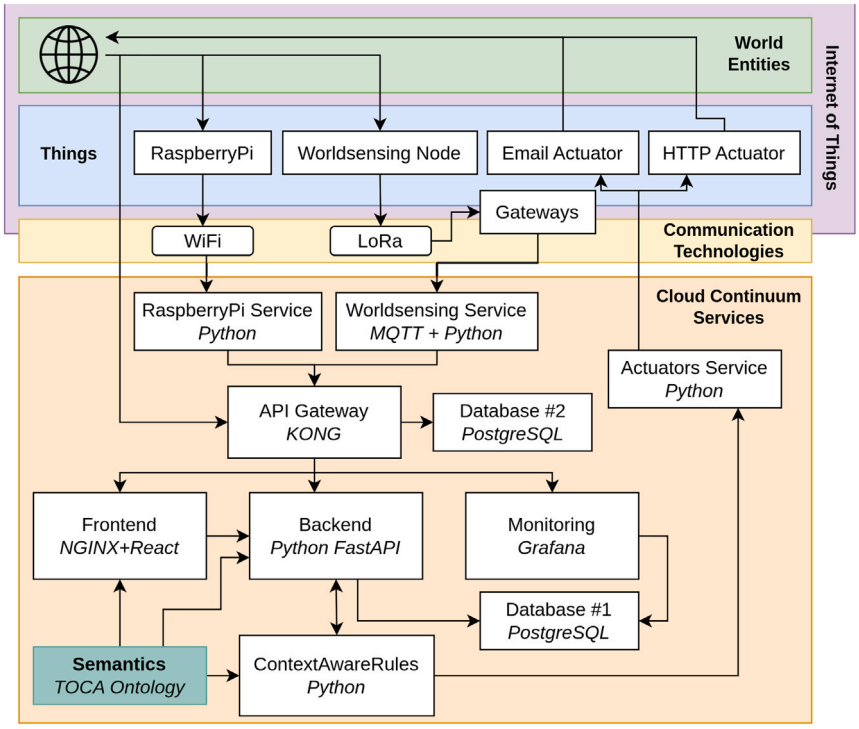


Fig. 9. Experimentation components mapped with our architecture proposal.

For example, we have developed endpoints to create new *Sensors* and to retrieve measurements (*Observations*), as well as all the conditions for providing context-aware services, such as the *EventRule*, *ConditionRule*, and *ResponseProcedure*.

We also include services related to the communication between the Things and the Cloud Continuum, these are included in the RaspberryPi service, which is responsible for collecting the information received through this sensor device and passing it to the entry point of the system; the Worldsensing service, which similarly collects the information sent by the Worldsensing node. In this case, as it uses LoRa as communication technology, the information is received from an intermediate point on the Internet, The Things Stack,[31] which is configured to send the information to our platform via an MQTT[32] broker, a lightweight publish/subscribe messaging transport protocol. In both cases, the implementation is done using Python programming language.

We have developed a context-aware engine that utilizes our ontology to automatically analyze incoming observations based on predefined events and conditions. Illustrated in Fig. 10, the engine is implemented in Python and is able to identify conditions that trigger the corresponding (re)actions, which are then executed using the Actuation service. The Actuation service supports *HTTP* and *email* communication protocols, enabling efficient and effective execution of actions. This implementation streamlines the process of analyzing observations, enabling quick and automated decision-making and execution of actions. Conceptually, by "simple rules" we understand the *ContextAwareRules* that are composed of a single *EventRule* and one or more *ResponseProcedures*. A rule becomes "complex" when it includes at least one *ConditionRule*.

Within its bounds, our ontology does support real-time action and monitoring. Currently done by polling the context-aware engine every minute, where *events* and *conditions* are routinely checked. Although, the configuration can be easily changed depending on the use case needs. We also recommend including a queuing system, such as MQTT, within the suggested architecture if higher performance is needed.

Our front-end consists of a Javascript framework called React.js,[33] where all the metadata of the system is observed. As a monitoring data visualization system, we use Grafana,[34] which can be configured to be adapted correctly to most of the IoT data types that we deal with. Summing up the components explained, we have a monitoring and visualization system with the data reported by the devices. We also provide in our front-end a visualization tool that shows in real-time entities instantiated in the system.

[31] https://www.thethingsindustries.com.
[32] https://mqtt.org.
[33] https://reactjs.org.
[34] https://grafana.com.

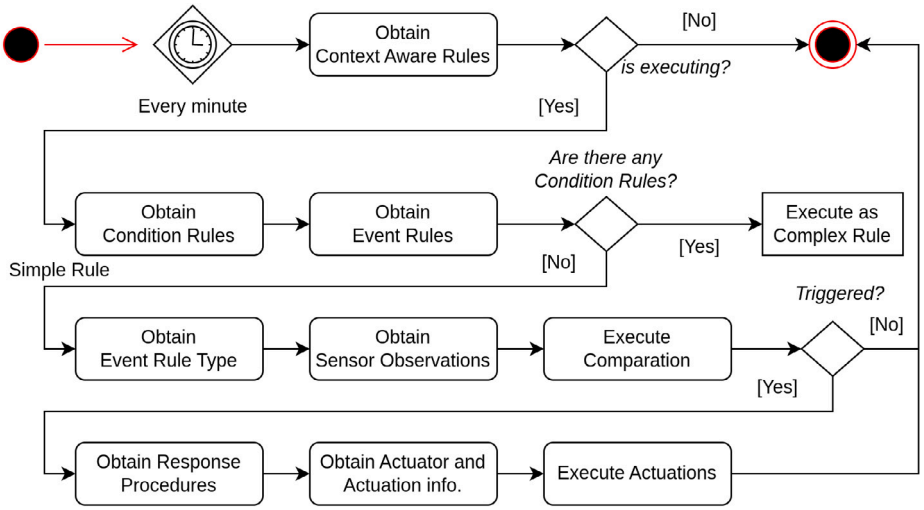


Fig. 10. UML Flow diagram stating the context-aware rule engine and the flow for a simple rule.

The whole system architecture uses PostgreSQL,[35] a relational database that can be optimized to handle data time series, as base data. Moreover, most of the experimentation components are microservices, orchestrated and deployed via Docker[36] and Docker-compose,[37] which also automates and accelerates the deployment process, resulting in faster and more efficient deployment of system components.

7.3. Experiment results

Our experiment aims to monitor the inclination of the railway tunnel wall. The *FeatureOfInterest* is a "RailwayTunnel", and the *ObservableProperty* to be monitored by the *Sensors* is the "WallInclination". To this end, we have the "GrovePiAccelerometer" hosted in the "RaspberryPi" *Thing* and the "InclinationSensor" contained within the "Worldsensing" *Thing*. The Raspberry Pi is taking measurements (*Observations*) of the wall inclination and sends them directly to the Cloud. A similar process is performed on the Worldsensing Node, where raw measurements are collected and sent using a LoRa radio, and the *Kerlink* gateway forwards the information to our Cloud server.

Our experiment has yielded the first validated objective, which demonstrates that the ontology design enables information exchange between different IoT devices and systems. Moreover, we have conducted several tests to verify that our ontology allows one to modify the information entered, thus, enabling data reception. In addition, it acts based on what the user has set up in the system. We have established a cloud infrastructure with the necessary software components to facilitate communication with our ontology. The platform operator has at his disposal the documentation to access all entities in the proposed system: the creation of entities, the consultation of entity information, and the deletion of entities. All these actions are performed through *HTTP REST calls*, with methods such as `GET`, `POST`, and `DELETE`.

The following steps outline one possible approach to configure the system. They guide the registration of the element to be monitored, along with the devices, rules, and actions to be checked and performed. Therefore, the system can effectively monitor the specified element and devices, check for specific rules, and perform the necessary actions based on the results:

1. Create the instance of the element to be monitored.
   (a) We create a *Location* named `Barcelona`.
   (b) A *FeatureOfInterest* named `TunnelWall`, that is in *Location* `Barcelona`.
   (c) An *ObservableProperty* named `Tilt`, which *TypeOfObservation* is that of an `integer`, pointing out to the already instanced `TunnelWall`.
2. Now, the system is ready to define the elements that are in charge of the monitoring. Once these steps are completed, the system status will be as shown in Fig. 11.
   (a) We create a *Group* named `OfficeBCN`, that is in *Location* `Barcelona`.

[35] https://www.postgresql.org.
[36] https://www.docker.com.
[37] https://docs.docker.com/compose/.

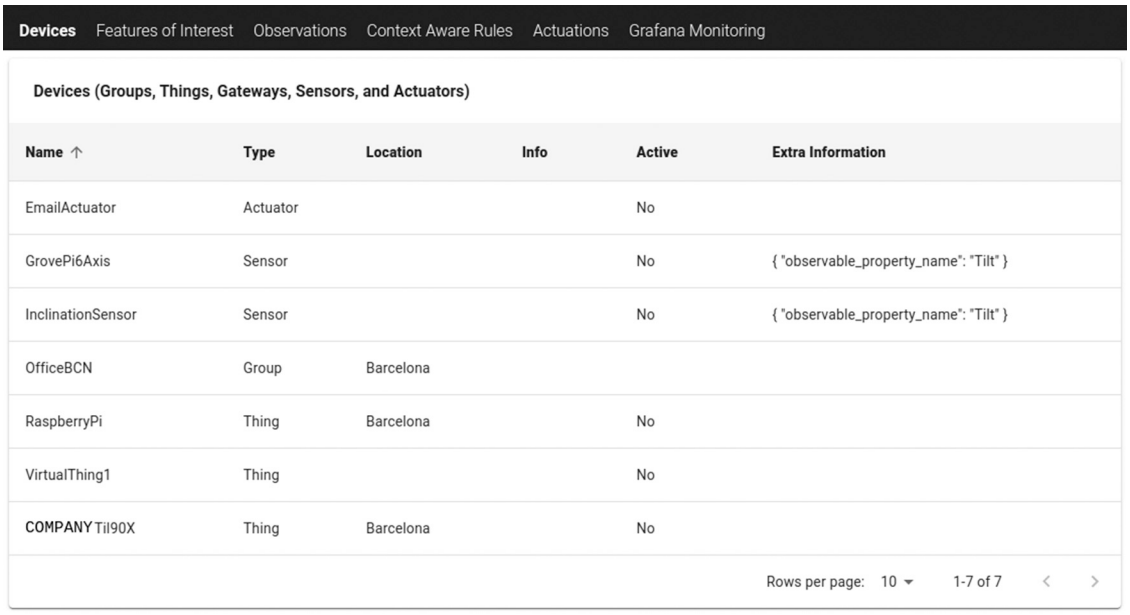


Fig. 11. Our Frontend showing the onboarded Groups, Gateways, Things, Sensors, and Actuators.

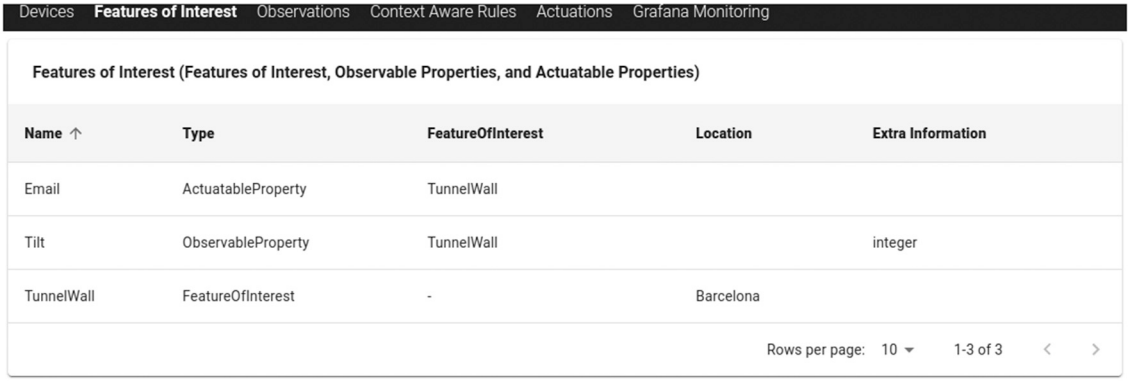


Fig. 12. Our Frontend showing the onboarded Features of interest, ObservableProperties, and ActuatableProperties.

(b) A *Thing* named `RaspberryPi`, that is in *Location* `Barcelona`.
(c) A *Sensor* named `GrovePi6Axis`, hosted by *Thing* `RaspberryPi`, observes the *ObservableProperty* `Tilt`, that is in *Location* `Barcelona`.
(d) A *Thing* named WorldsensingTil90X , that is in *Location* `Barcelona`.
(e) A *Sensor* named `InclinationSensor`, hosted by *Thing* WorldsensingTil90X , observes the *ObservableProperty* `Tilt`, that is in *Location* `Barcelona`.

3. Once the necessary information has been gathered to handle *Observations*, the next step is to onboard entities for *Actuation*. Once these steps have been completed, the status of the system will be reflected as shown in Fig. 12.

(a) We assume the same *Location*, `Barcelona`, and the same *FeatureOfInterest* named `TunnelWall`.
(b) We create an *ActuatableProperty* named `Email`, pointing out to the `TunnelWall` element.
(c) A *Thing* named `VirtualThing1`, with no *Location* since it is a virtual entity that will handle *Actuations*.
(d) We create an *Actuator* named `EmailActuator`, pointing out to the already created *Thing VirtualThing1* and *ActuatableProperty* `Email`.

4. Lastly, we onboard all the required elements to enable the functioning of *ContextAwareRules*. Once these steps are completed, the system is set up as illustrated in Fig. 13.

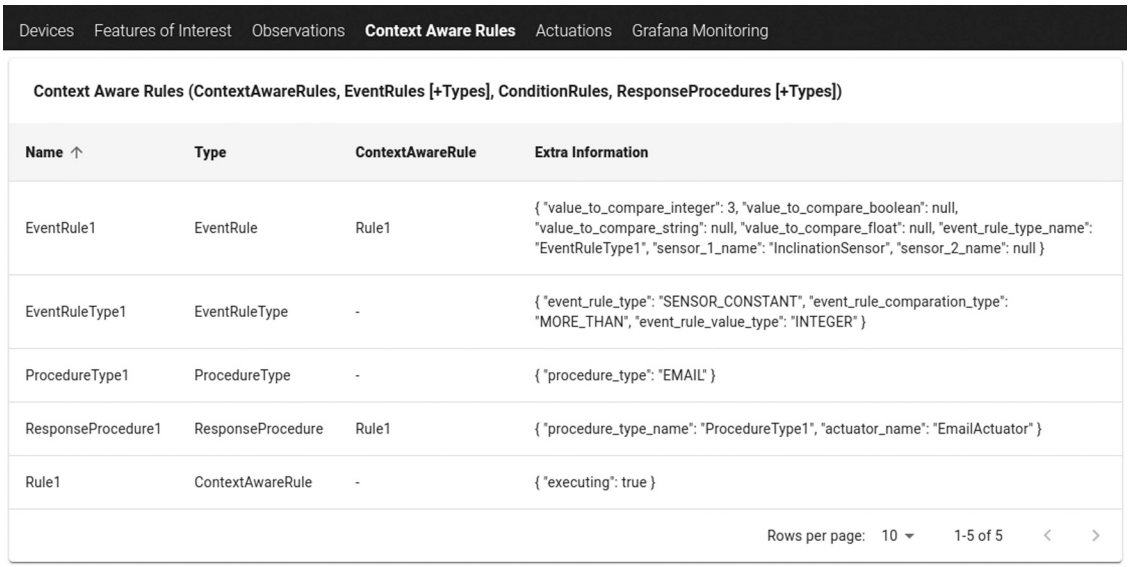


Fig. 13. Our Frontend showing the onboarded rules and their components.

(a) We create a *ContextAwareRule*, named `Rule1` and *executing* as *true*.
(b) An *EventRuleType*, named `EventRuleType1`, that its *type* is `SENSOR_CONSTANT`. The *ComparationType* is `MORE_THAN`. And the *ValueType* is `Integer`. This all means that it compares a *Sensor* value is greater than a constant value, both of which are `integers`.
(c) An *EventRule*, named `EventRule1`, for the *ContextAwareRule* `Rule1`, with the `EventRuleType1` type, comparing the *Sensor* `InclinationSensor` and the *value_integer* 3.
(d) A *ProcedureType* named `ProcedureType1` which *type* is that of sending an `email`.
(e) A *ResponseProcedure*, named `ResponseProcedure1` for the *ContextAwareRule* `Rule1`, with the *Type* of `ProcedureType1` and *Actuator* named `EmailActuator`.

Once these steps are executed, then the system is properly configured to receive measurements. Whenever that ocurrs, the system will generate an *Observation*, as in Listing 1 including the reading value in the *value* field, as well as the time of observation. As defined in the ontology, the observation time is an extension of the *time:inXSDDateTimeStamp* (class *Time:Instant*) from the OWL-Time ontology.

Listing 1: JSON that the Raspberry Pi sends for creating an Observation

```
POST /observations/
Body: { "sensor_name": "InclinationSensor",
  "time_start": "2023-04-08T14:27:04+00:00",
  "observable_property_name": "Tilt",
  "value_int": 28 }
```

By creating an *Observation* for each gathered measurement, the system can effectively manage the collected data and provide relevant information to end-users. Observation *time* and the *Sensor* plays a crucial role in ensuring that data is properly contextualized, to later identify patterns, trends, and anomalies in the data, providing valuable insights for various applications, such as predictive maintenance and environmental monitoring. Fig. 14 shows the *Observations* tab in our experimentation. On the left, we show an updated list of these *Observations*, while the right side displays them visually through an instance of Grafana. This approach enables the provision of information in a faster and more visual manner.

The *Observations* sent by the *Thing* connected via WiFi are fed into the system almost directly, as the communication technology itself allows direct calls to be sent using the API of our Cloud server. On the other hand, *Observations* sent by the Worldsensing node are sent via radio (LoRa), and the gateways in coverage retransmit them to a middleware software, The Things Stack. As shown in Fig. 15, the middleware receives data from our device, the `Thing:WorldsensingTil90X` , which is illustrated in the figure as *ls-41329*, its public *label*.

As the LoRa communication technology is not point-to-point, data can flow through different gateways, so the information is transmitted in a specific format. Therefore, for each packet, in addition to the observation information, which is included in the `frm_payload` field, the following data are sent, including the signal metadata, the communication itself, and some others. These are the elements we previously mentioned in the ontology definition. They refer to the auxiliary data that are transmitted

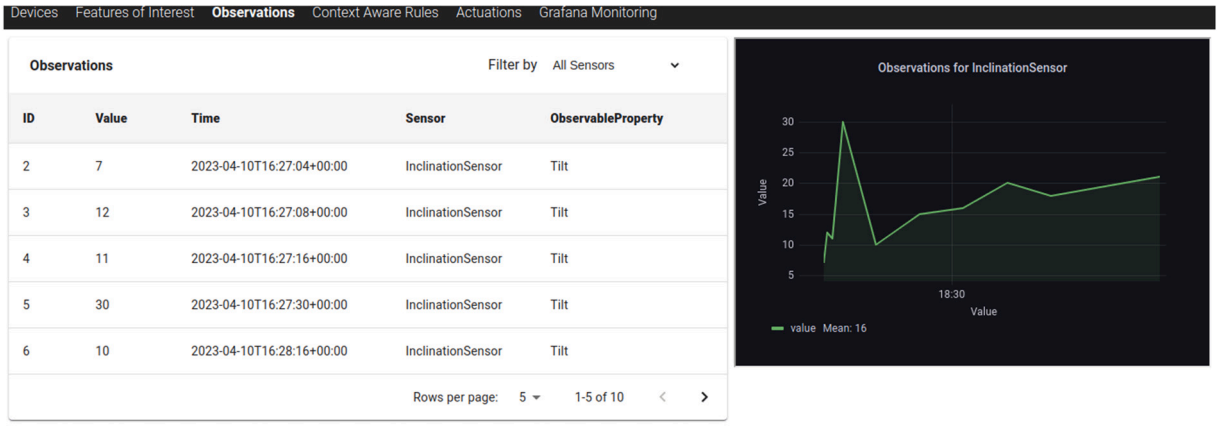


Fig. 14. Our Frontend showing the Observations received.

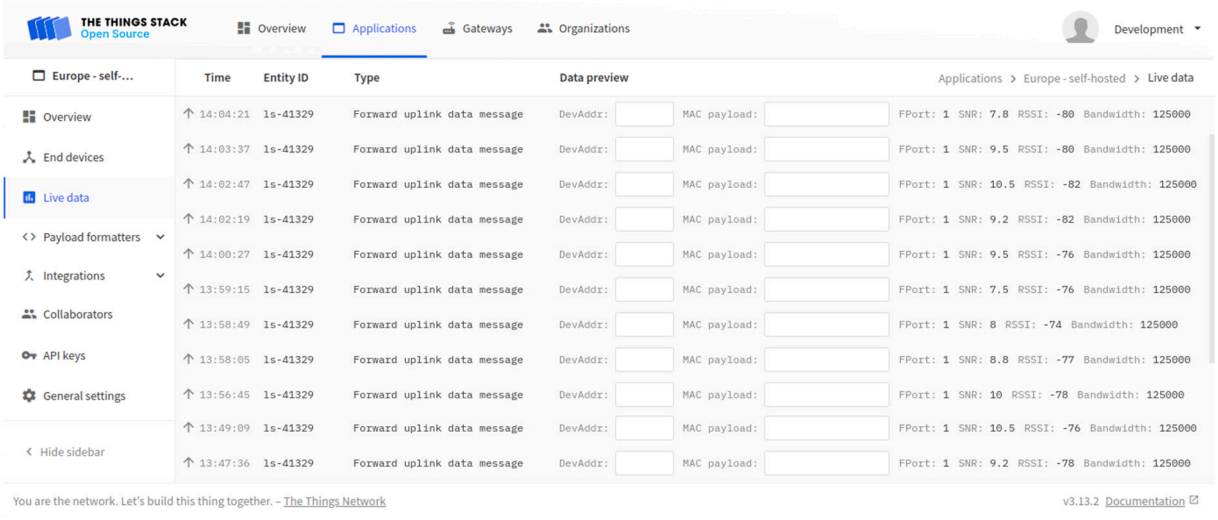


Fig. 15. The Things Stack dashboard showing the events received by the Tiltmeter IoT node.

alongside certain communication technologies and can also be classified as data from sensors. A summary of the elements sent in each node transmission and forwarded by *The Things Stack MQTT Broker* can be seen in Listing 2.[38]

Listing 2: JSON that the Tiltmeter IoT node sends for creating an Observation

```
{  "end_device_ids": {
      "device_id": "ls-41329",
      "application_ids": { "application_id":"HIDDEN" },
      "dev_eui": "HIDDEN", "join_eui": "HIDDEN", "dev_addr": "HIDDEN"
   },
   "correlation_ids": [
      "as:up:HIDDEN", "gs:conn:HIDDEN", "gs:up:host:HIDDEN", "gs:uplink:HIDDEN", "ns:uplink:HIDDEN", "
          rpc:/ttn.lorawan.v3.GsNs/HandleUplink:HIDDEN", "rpc:/ttn.lorawan.v3.NsAs/HandleUplink:HIDDEN
          "
   ],
   "received_at": "2023-04-07T08:06:35.581094134Z",
   "uplink_message": {
      "session_key_id": "HIDDEN",
      "f_port": 1, "f_cnt": 19493,
      # THE ACTUAL DATA VALUE PROVIDED BY WS NODE
      "frm_payload": "HIDDEN",
```

[38] References for each field are in The Things Industries documentation: https://www.thethingsindustries.com/docs/reference/data-formats/.

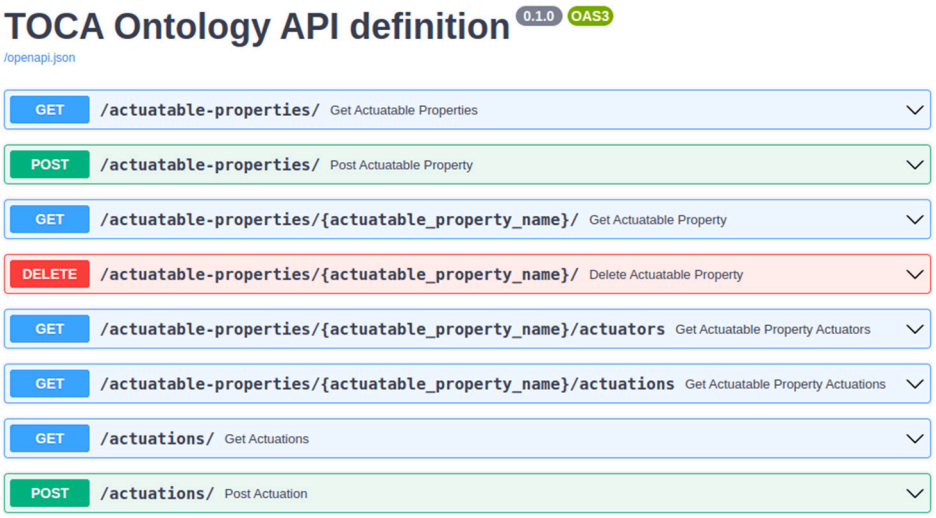


Fig. 16. Part of the TOCA ontology and its OpenAPI description[20].

```
      "rx_metadata": [
         {
            "gateway_ids": { "gateway_id": "ls-gw-25681", "eui": "HIDDEN" },
            "timestamp": 188100452,
            "rssi": -80, "channel_rssi": -80,
            "snr": 8.2, "channel_index": 5, "uplink_token": "HIDDEN"
         }
      ],
      "settings": {
         "data_rate": { "lora": { "bandwidth": 125000, "spreading_factor": 7 } },
         "data_rate_index": 5, "coding_rate": "4/5",
         "frequency": "867500000",
         "timestamp": 188100452
      },
      "received_at": "2023-04-07T08:06:35.353710286Z",
      "consumed_airtime": "0.071936s" }}
```

Our experimentation involves the development of various software concepts, accessible through API calls, which we include altogether with their corresponding schema in OpenAPI format, as illustrated in a summarized way in Fig. 16. OpenAPI (formerly known as Swagger) is a widely-used specification for building and documenting APIs, providing a standardized way to describe the functionality of web services. An example of this documentation is found in Listing 1. In total, to follow the steps indicated above and perform the onboarding of all entities, 17 similar operations to those on Listing 1 have been executed, which we do not show for the sake of simplicity.

The complete experiment's working code is available at https://github.com/worldsensing/critical-infrastructure-awareness-internet-of-things-context-data. This code serves as an orchestration and management tool for the entire architecture described in Section 7.1. Additionally, it includes the code responsible for the context-aware rules engine. In terms of data measurements, the code handles the Raspberry Pi's physical sensor data transfer to the cloud backend, as well as that from the Worldsensing node.

8. Conclusions and future work

In this article, we have presented TOCA, an ontology for monitoring world entities by using IoT devices. We describe the concepts of monitoring in general, from observation to action, including the definition of context rules, to act accordingly when the situation requires so. With our proposal, we enable end users to develop their own IoT use case, as we provide with the conceptual and experimentation part of the proposal.

Our ontology is based on known ontologies. It has as main points a correct definition of the entities that make possible the observation and actuation of the Internet of Things elements, enabling the modeling of the necessary entities to have information about the devices employed in the monitoring, including a novel way to define gateways entities. We also include the definition of the entities that allow performing actions when certain conditions are met, emphasizing the context-awareness concept. Additionally,

[20] Full OpenAPI definition is located at the `./backend/openapi_backup.json` in our provided GitHub repository

we provide mechanisms to develop conceptual models for use cases in the broad Cloud Continuum paradigm, including allowing for the distribution of the application components among the different paradigm tiers.

We propose a general architecture for the IoT paradigm, with the main goal to improve clarity when thinking about IoT systems among researchers. In our proposal, we conceptualize and define the elements that are monitored, the elements that are used for that purpose, the communication technologies that carry the information, as well as the supporting cloud software system that is responsible for handling all the required information for a correct monitoring scenario.

Alongside this, we have also proposed an experimentation that instantiates our proposed ontology and architecture. The domain is on the railway, a type of critical infrastructure, where monitoring is key for the correct and safe operation of the railway. For this experimentation, we use a prototype device, a Raspberry Pi, and also a Worldsensing production device. The objective is to monitor the inclination of a railway tunnel wall.

Further work is needed in general in the area of interoperability of things. In this article, we show the potential and benefits of semantics in the IoT domain, and, in particular, in the domain of monitoring and context awareness. However, we have some points of improvement at the level of the experimental code, where we have made some simplifications like finalizing the process of reviewing the rules when they are complex, as well as implementing services that actually send an email, or even make a phone call. We also plan to incorporate the concepts explained here in the operational part of Worldsensing as well as to incorporate the Machine Learning path to perform predictive maintenance using real-time data.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

The data used is available through a link in the article itself

Acknowledgments

This work is partially funded by Industrial Doctorates from Generalitat de Catalunya (2019 DI 001), the SUDOQU project, PID2021-126436OB-C21 from MCIN/AEI,10.13039/501100011033, FEDER, UE, and the Grup de Recerca Consolidat IMP, 2021-SGR-01252. We also thank the reviewers for their valuable comments.

References

[1] C. Perera, A. Zaslavsky, et al., Context aware computing for the internet of things: A survey, IEEE Commun. Surv. Tutor. 16 (1) (2014) 414–454.
[2] L. Bittencourt, R. Immich, et al., The internet of things, fog and cloud continuum: Integration and challenges, Internet Things 3–4 (2018) 134–155.
[3] F. Xhafa, B. Kilic, P. Krause, Evaluation of IoT stream processing at edge computing layer for semantic data enrichment, Future Gener. Comput. Syst. 105 (2020) 730–736.
[4] A.A. Alli, M.M. Alam, The fog cloud of things: A survey on concepts, architecture, standards, tools, and applications, Internet Things 9 (2020) 100177.
[5] N. Paudel, R.C. Neupane, A general architecture for a real-time monitoring system based on the internet of things, Internet Things 14 (2021) 100367.
[6] M. Elkhodr, S. Shahrestani, H. Cheung, The internet of things: New interoperability, management and security challenges, Int. J. Netw. Secur. Appl. 8 (2) (2016) 85–102.
[7] E. Lee, Y.-D. Seo, et al., A survey on standards for interoperability and security in the internet of things, IEEE Commun. Surv. Tutor. 23 (2) (2021) 1020–1047.
[8] M. Ganzha, M. Paprzycki, et al., Semantic interoperability in the internet of things: An overview from the INTER-IoT perspective, J. Netw. Comput. Appl. 81 (2017) 111–124.
[9] N. Widell, A. Keränen, R. Badrinath, What is Semantic Interoperability in IoT and Why Is It Important?Technical Report, Ericsson, 2020.
[10] Alliance for IoT and Edge Computing Innovation, Standardisation, 2022, URL https://aioti.eu/resources/standardisation-resources/.
[11] M. Noura, M. Atiquzzaman, et al., Interoperability in internet of things: Taxonomies and open challenges, Mob. Netw. Appl. 24 (2019) 796–809.
[12] N.F. Noy, D.L. McGuiness, Ontology Development 101: A Guide to Creating Your First Ontology, Technical Report, Knowledge Systems - Stanford University, 2001.
[13] T. Bittner, M. Donnelly, S. Winter, Ontology and semantic interoperability, in: Large-Scale3D Data Integration, CRC Press, 2006, pp. 139–160.
[14] R. Jasper, M. Uschold, A framework for understanding and classifying ontology applications, in: Proc. IJCAI99 Workshop on Ontologies and Problem-Solving Methods, 1999.
[15] M.E. Jennex, Big data, the internet of things, and the revised knowledge pyramid, SIGMIS Database 48 (4) (2017) 69–79.
[16] R.L. Ackoff, From data to wisdom, J. Appl. Syst. Anal. 16 (1) (1989) 3–9.
[17] C. A. Tokognon, B. Gao, et al., Structural health monitoring framework based on internet of things: A survey, IEEE Internet Things J. 4 (3) (2017) 619–635.
[18] F. Lamonaca, P. Sciammarella, et al., Internet of things for structural health monitoring, in: 2018 Workshop on Metrology for Industry 4.0 and IoT, 2018, pp. 95–100.
[19] S. Geetha, S. Gouthami, Internet of things enabled real time water quality monitoring system, Smart Water 2 (2017) 1.
[20] T. Malche, P. Maheshwary, R. Kumar, Environmental monitoring system for smart city based on secure internet of things (IoT) architecture, Wirel. Pers. Commun. (2019).
[21] A.S. Pillai, G.S. Chandraprasad, et al., A service oriented IoT architecture for disaster preparedness and forecasting system, Internet Things 14 (2021) 100076.
[22] C. Choi, C. Esposito, et al., Intelligent power equipment management based on distributed context-aware inference in smart cities, IEEE Commun. Mag. 56 (7) (2018) 212–217.

[23] A. Alsaig, V. Alagar, S. Nematollaah, Contelog: A declarative language for modeling and reasoning with contextual knowledge, Knowl.-Based Syst. 207 (2020) 106403.
[24] C. Perera, A. Zaslavsky, et al., CA4iot: Context awareness for internet of things, in: IEEE International Conference on Green Computing and Communications, 2012, pp. 775–782.
[25] V. Alagar, M. Mubarak, et al., A framework for developing context-aware systems, EAI Endorsed Trans. Context-Aware Syst. Appl. 1 (1) (2014) e2.
[26] V.A. Shekhovtsov, S. Ranasinghe, et al., Domain specific models as system links, in: Advances in Conceptual Modeling, 2018, pp. 330–340.
[27] G. Kim, S. Kang, et al., An MQTT-based context-aware autonomous system in onem2m architecture, IEEE Internet Things J. 6 (5) (2019) 8519–8528.
[28] I. Uddin, A. Rakib, H. Haque, P.C. Vinh, Modeling and reasoning about preference-based context-aware agents over heterogeneous knowledge sources, Mob. Netw. Appl. 23 (2018) 13–26.
[29] R. Dobrescu, D. Merezeanu, S. Mocanu, Context-aware control and monitoring system with IoT and cloud support, Comput. Electron. Agric. 160 (2019) 91–99.
[30] A.F. da Silva, R.L. Ohta, et al., A cloud-based architecture for the internet of things targeting industrial devices remote monitoring and control, IFAC-PapersOnLine 49 (30) (2016) 108–113.
[31] G.M. Lunardi, F.A. Machot, V.A. Shekhovtsov, et al., IoT-based human action prediction and support, Internet Things 3–4 (2018) 52–68.
[32] P. Cong-Vinh, Formal aspects of self-in autonomic networked computing systems, in: Autonomic Computing and Networking, Springer US, 2009, pp. 381–410.
[33] A. Haller, K. Janowicz, et al., The modular SSN ontology: A joint W3C and OGC standard specifying the semantics of sensors, observations, sampling, and actuation, Semant. Web J. (2018).
[34] K. Janowicz, A. Haller, et al., SOSA: A lightweight ontology for sensors, observations, samples, and actuators, J. Web Semant. 56 (2019) 1–10.
[35] A. Rhayem, M.B.A. Mhiri, F. Gargouri, Semantic web technologies for the internet of things: Systematic literature review, Internet Things 11 (2020) 100206.
[36] X. Mountrouidou, B. Billings, L. Mejia-Ricart, Not just another internet of things taxonomy: A method for validation of taxonomies, Internet Things 6 (2019) 100049.
[37] J. Aguilar, M. Jerez, T. Rodríguez, Cameonto: Context awareness meta ontology modeling, Appl. Comput. Inform. 14 (2) (2018) 202–213.
[38] C. Angsuchotmetee, R. Chbeir, Y. Cardinale, MSSN-onto: An ontology-based approach for flexible event processing in multimedia sensor networks, Future Gener. Comput. Syst. 108 (2020) 1140–1158.
[39] G. Privat, Guidelines for Modelling with NGSI-LD, Technical Report, ETSI, 2021.
[40] J. Kiljander, A. D'elia, et al., Semantic interoperability architecture for pervasive computing and internet of things, IEEE Access 2 (2014) 856–873.
[41] M. Vila, M.-R. Sancho, E. Teniente, Modeling context-aware events and responses in an iot environment, in: Advanced Information Systems Engineering, in: CAiSE, 2023, pp. 71–87.
[42] M. Vila, V. Casamayor, S. Dustdar, E. Teniente, Edge-to-cloud sensing and actuation semantics in the industrial internet of things, Pervasive Mob. Comput. 87 (2022) 101699.
[43] OGC, GeoSPARQL - a geographic query language for RDF data, 2012, URL https://www.ogc.org/standards/geosparql, [Last accessed 05 Feb 2023].
[44] W3C - OWL-Time, Time ontology in OWL, 2020, URL https://www.w3.org/TR/owl-time/, [Last accessed 06 Feb 2023].