Modeling Context-Aware Events and Responses in an IoT Environment

Marc Vila[1,2(✉)], Maria-Ribera Sancho[1,3], and Ernest Teniente[1]

[1] inLab FIB, Universitat Politècnica de Catalunya, Barcelona, Spain
{marc.vila.gomez,maria.ribera.sancho,ernest.teniente}@upc.edu
[2] Worldsensing, Barcelona, Spain
[3] Barcelona Supercomputing Center, Barcelona, Spain

Abstract. The Internet of Things (IoT) involves the use of devices that exchange information about the state of things in the real world. In IoT, monitoring is regarded to be the most fully researched use case. However, research on the use and manipulation of control and maintenance applications has not yet been fully addressed. An important step forward in this direction may be provided by executing automatic context-aware actuations. These may be achieved by delivering responses based on the context gathered with components endowed in some device. In this paper, we propose a solution that uses ontological knowledge for this purpose, thus improving the interoperability of IoT devices. We focus on real-time data collection to fully automate monitoring, context gathering, and appropriate responses. Our proposal is illustrated via the lens of a railroad use case, where maintaining track safety is critical to avoid accidents.

Keywords: Internet of Things · Interoperability · Context-Awareness · Semantics · Cyber-Physical Systems

1 Introduction

The Internet of Things (IoT) comprises a large number of *smart* devices: physical objects aimed at connecting and exchanging information with other devices, entities, or systems via the Internet. IoT sensors generate data that can be used for various purposes, such as monitoring, data analysis, or decision-making. IoT actuating devices are typically used to convert a signal input into a physical action or movement or to modify the logical state of an entity.

Physical infrastructures must be effectively monitored to ensure their reliability, security, and performance, and to detect and resolve any possible difficulties or problems. There are situations where entities, whether real or virtual, need to be watched over and, if necessary, reacted to. When an informational condition has to be taken into account, a responsive capacity is required; if it is urgent, it has to be handled accordingly. This situational context is mostly gained by employing IoT devices with sensors and logical entities, such as data transmission information, depending on whether the state can be inferred from the outside

M. Indulska et al. (Eds.): CAiSE 2023, LNCS 13901, pp. 71–87, 2023.
https://doi.org/10.1007/978-3-031-34560-9_5

world or from any value associated to software. This knowledge or ability, namely "context-aware", involves considering the data that the IoT devices gather and allowing the system to understand its context. The system can then react since it is aware of the situation where it is. Context awareness is a key aspect of the IoT domain that enables systems to provide a more precise understanding of the environment [24]. As a result, they can provide accurate responses to events. For example, context-aware situations can be used in a factory to monitor machinery performance and optimize production by considering factors such as temperature, humidity, and energy consumption.

In IoT, communication is straightforward when the systems are in the same working environment. However, with entities from other companies or platforms, communication becomes more complicated. This complexity is due to the fact that two different environments must agree to establish communication. This has implications for information properties, data structures, and communication technologies. The *Interoperability of Things* aims to homogenize data communication between IoT devices so that software systems can provide generic solutions to enable interoperability across applications, contexts, and domains [9].

In this paper, we contribute to the Interoperability of Things as follows:

- We propose an ontology for automatic monitoring using IoT devices. We focus on the definition of context-aware entities that enable to specify the responsive behavior to be taken when a certain situation is met. We allow for defining the entities to be observed, receiving the measurement data from the sensors, and also specifying the actions or response procedures to be taken when a given event occurs.
- Our ontology is domain-independent, and it is based on existing ontologies such as SSN/SOSA, GeoSPARQL, OWL-Time, and IoTMA. All software systems dealing with its components are able to handle different IoT installations with almost no changes in the code. A small number of components might be needed to update, for instance, how the various devices feed information to the ontology or extend actuation procedures.
- We provide an implementation based on a rail-track safety monitoring scenario, to show the feasibility of our approach.

Section 2 reviews related work. Section 3 highlights the key elements of the ontology and compares them with previous work. Section 4 depicts our use case and matches the proposed ontology with it. Section 5 shows the experimentation carried out. The paper ends with our conclusions and further work.

2 Related Work

The term *IoT* was originally coined in 1999 for supply chain management by Ashton [5] and was later used to encompass the devices aimed at exchanging information with other elements on the Internet [11]. One of the most notorious characteristics of IoT is the heterogeneity of the ecosystem, i.e. the *Interoperability of Things*, one of the challenges to be solved. There is heterogeneity in

the devices, not only from differences in capacity and features but also for application requirements and information transmission [26]. This can also be seen in the technologies used to communicate and in the information data structure.

According to Noura et al. [19], interoperability in IoT can be classified into different levels. The *device* level is related to output capacity and communication protocols; the *syntactic* one to the data format, schemas, and interfaces; *networking* to the network protocols; *platform* to the operating system, and programming language; and *semantic* to the data and information models. Our work aims at improving the interoperability of IoT devices at the semantic level in the context-awareness domain. This allows for the abstraction of the particular syntax and data formats, providing common semantics to all the managed data.

Context-awareness is known as the ability of software applications to discover and react to changes in the environment in which they are situated. In the IoT setting, this concept enables the contextualization of the information linked to sensor data so that interpretations can be easily and meaningfully performed, as stated Schilit et al. [27] in 1994. Some years later, Abowd et al. [1] provided a more precise definition: "A system is context-aware if it uses context to provide relevant information and/or services to the user, where relevancy depends on the user's task". In 2004, contextual information began to be combined with ontologies [13,28,31], in which human, machine, physical, and abstract things are combined, using ontologies, modeling, and semantic reasoning.

In the general scope of context-awareness in the IoT, Perera et al. [23] proposed an IoT context-aware architecture with the purpose of automatically selecting sensor data from some user-based inputs. However, they focus on the ability to select the most significant sensor for a particular task and do not provide a middleware solution for managing its context. Kim et al. [17] developed a system to support an autonomous context-aware environment where they proposed low complexity rules to execute *if-else* kind of rules. These rules are useful in certain restricted contexts but they are not general enough since they do not allow combining sensors for instance. Dobrescu et al. [7] proposed a middleware architecture for IoT context-aware field monitoring using environmental and real-time sensors. However, it is not clear how they support the use of different sensors beyond the ones they already considered. Jiang et al. [16] stated that context-awareness with semantics is a recent research area, and sketched an ontology for the integration of several data sources to extract contextual information. Gaur et al. [10] proposed a context-aware proof-of-concept framework, but it is not clear how context actions can be incorporated and handled. In Zhang et al. [33], authors propose an approach to capture events from sensor streams. However, it is not clear how the data is structured in the sense of heterogeneity of things. Dörndorfer et al. [8], developed a modeling tool that includes data aggregation and definition for sensors, context, and decision outputs. It is not clear how multiple context-aware rules can be defined, as it seems to remain in the *if-else* theory.

Improving interoperability has also been achieved through ontology definition. Therefore, the Semantic Sensor Network Incubator Group developed the

SSN [12] ontology in 2011 to specify sensors and sensor network resources. In 2017, the W3C Consortium proposed the SOSA [15] ontology, an extension of SSN, to be used when the Semantic Web and linked data technologies are needed. Xue et al. [32] proposed a semantic sensor network but the proposal is not complete as far as managing the network. Maarala et al. [18], examine the different types of semantic reasoning and different data models for context-aware IoT applications. They also propose an ontology for information contextualization. However, they do not provide means to understand the sensing part of the process. Alirezaie et al. [3] and Choi et al. [6] propose a context-aware system for smart homes and cities. CAMeOnto [2] proposes a meta-ontology for modeling context-awareness systems, including *user, activity, time, device, services,* and *location*. Yet, they do not take into account the sensorization part, and the relationship between what is being observed and what is being controlled or acted upon is not clear. MSSN-Onto [4] proposes an ontology that models sensors, events, and their corresponding types. They use SOSA/SSN as a base ontology, but do not go beyond its observation perspective.

In conclusion, context awareness with semantics and ontologies is a hot topic that only recently began to receive significant attention. Previous work mostly uses only a subset of features, among all those encompassed by the IoT. Moreover, finding proposals like ours that successfully combine context awareness systems, ontologies for the IoT, and the modeling of sensors and actuators is still a challenging issue.

3 Our Context-Aware Responsive Ontology

The ontology we present in this paper is aimed at enabling the monitoring of real-world entities using IoT devices in computing continuum applications. The context-aware responsive behavior of our ontology is based on two different concepts: *Actuation* and *ContextAwareRule*. Context-aware rules specify the general policies that will be applied when a certain situation is satisfied, such as when a sensor detects a value that is higher than a certain threshold and takes some action. These are the *Actuations*, which specify the specific corrections or actions made after an event. In our approach, the designer can precisely specify the events to be monitored, under which conditions, and the actions to be taken when a condition is satisfied.

Our ontology, Context-Aware Responsive ontology (ConAwarIoT) extends the IoTMA ontology [29] by incorporating the specification of context-aware events and responses to these events, which were not considered. Our proposal includes two key concepts from IoTMA: *Sensor*, the element capable of making measurements of a given feature; and *Actuator*, the executors of actions that modify the state of a given feature. We extend these concepts here with the new key concept *ContextAwareRule*, which allows users to specify the events to monitor, conditions to apply and response procedures to execute.

Our ontology is defined in Fig. 1 by means of a UML class diagram, showing the *classnames* and the relations between *classes*:

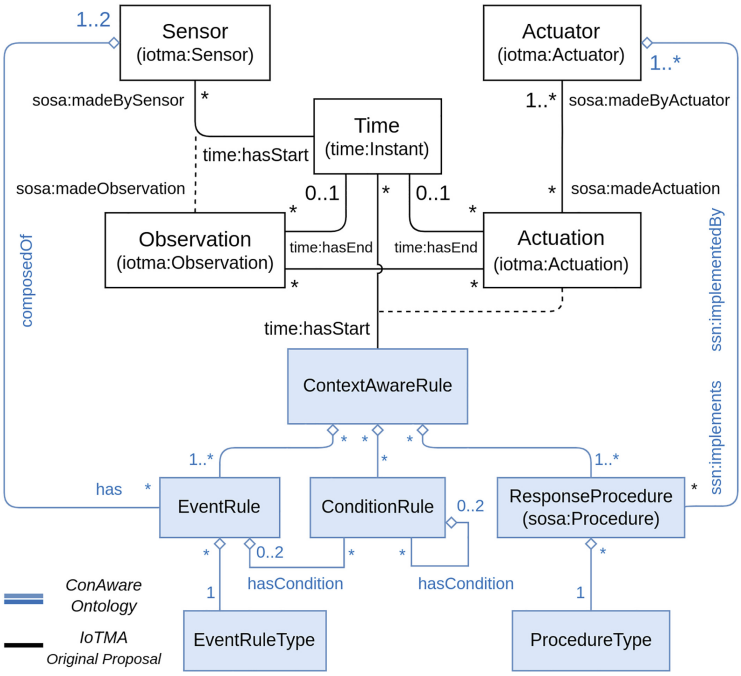


Fig. 1. Overview of our Context-Aware Responsive ontology

We share with IoTMA the concepts *Sensor*, *Actuator*, *Observation*, *Actuation*, and *Time*. This is because we want to stress that the definition and treatment of responsive behavior require this information to be effective. In the previous figure, ConAwarIoT is the default namespace when none is provided.

3.1 Resource URIs

In our work, *Classnames* are expressed in the Uniform Resource Identifier (URI) format, a unique sequence of characters that identifies a resource, to increase the simplicity and manageability of systems. In Table 1 we show the URI additions defined in our ontology. The first column represents the *classname* of the entity. The second is the URI, the format that follows the pattern defined below. `BASE_URI` refers to the URL entry point located in a test domain. `CLASS_NAME` indicates the name of the entity. `CLASS_ID` the identifier for each instance of an entity. In addition to `CLASS_PROPERTY_N`, which holds N properties of the entity: `{BASE_URI}:{CLASS_NAME}/{CLASS_ID}?{CLASS_PROPERTY_1}&{CLASS_PROPERTY_N}`

3.2 Description of the Concepts for Our ConAwarIoT

ContextAwareRule. It states the rules or conditions that define how the system reacts when some condition is met. If so, an *ResponseProcedure* is executed

Table 1. Main URIs added in our Context-Aware Responsive Ontology

Class	URI Patterns
ContextAwareRule	`ConAwarIoT:ContextAwareRule/{ContextAwareRuleName}?{EventRulesName}&{ConditionRulesName}&{ResponseProceduresName}&{Executing}`
EventRule	`ConAwarIoT:EventRule/{EventRuleName}?{ContextAwareRuleName}&{EventRuleTypeName}&{Sensor1Name}&{Sensor2Name}&{ValueBoolean|ValueString|ValueInteger|ValueFloat}`
ConditionRule	`ConAwarIoT:ConditionRule/{ConditionRuleName}?{ContextAwareRuleName}&{EventRule1Name}&{EventRule2Name}&{ConditionRule1Name}&{ConditionRule2Name}&{ConditionComparationType}`
ResponseProcedure	`ConAwarIoT:ResponseProcedure/{ResponseProcedureName}?{ContextAwareRuleName}&{ProcedureTypeName}&{ActuatorName}`

via an *Actuator*'s *Actuation*. Context-aware rules hold metadata information about sensor-actuator mechanisms and are made up of events (*EventRule*), conditions (*ConditionRule*), and responses (*ResponseProcedure*). It is also possible to state which sensors should be taken into account, for what reason, and what to do when *that* occurs. In addition, these rules can be toggled, enabled, or disabled, to enable all event detection and responses using the *executing* attribute.

A simple example could be, when a certain EventRule is true, it executes a certain ResponseProcedure action as a response, as can be seen in Fig. 2.

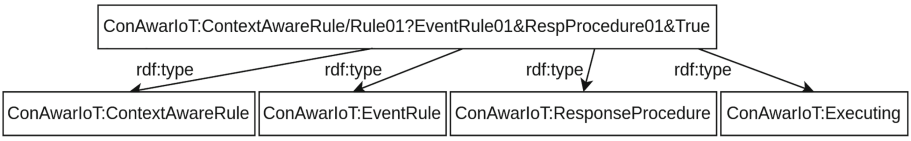


Fig. 2. Example of a ContextAwareRule description

EventRule. It is used to check whether a sensor measurement complies with a predefined criterion. Starting from a *Sensor* or a set of sensors, it allows defining in *EventRuleType* which comparison types should be made, either one sensor compared to a constant (*Value*) or with another sensor (*Sensor*). In *EventRuleType*, we assume the following comparison operators (*EventRuleComparationType*) available to compare among *Sensors* or against constant values:

- `EQUALS` or `NOT_EQUALS`: Numeric values such as `INTEGER` or `FLOAT`, also `STRING` and `BOOLEAN`.
- `LESS_THAN` or `MORE_THAN`: Numeric values such as `INTEGER` or `FLOAT`.
- Mathematical operators such as the *arithmetic mean*, *median*, *harmonic mean* or the *standard deviation*.

As an example, Fig. 3 represents an *EventRule*, named `EventRule01` that compares a sensor `Sensor01` against a constant `ValueInt1`. The comparison is about two `integers`, and it is desired to know if the contents are `equal` or not. And it could be similar to the comparison between a temperature sensor that is being compared over being equal to a value, for example, 20 °C.

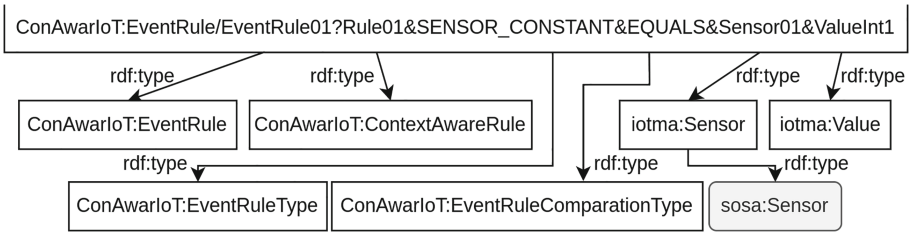


Fig. 3. Example of an EventRule description, comparing one sensor - one constant

ConditionRule. *EventRules* can be combined using *ConditionRules*. This is required when two or more entities in the real world have to be checked together. The operators we are able to deal with are `AND`, `OR`, `NAND`, `NOR`, and `XOR`. Moreover, *ConditionRules* can be linked, as components of the condition itself, to take advantage of the possibility of specifying different conditions that handle *EventRules*. Two side operators are available: one operator to be *EventRule* and one *ConditionRule*, as well as a couple of *EventRule* or a couple *ConditionRule*. If more complex *Rule* are needed, they can be nested using this concept.

ResponseProcedure. Handles the definition of actions to be executed when defined criteria are met. It contains the *Procedure* to follow and the predefined steps to improve in the target scenario. It also allows the user to state which type of actuation is needed, for instance, the *HTTP + GET* method in Fig. 4, which means that when the *Actuation* is executed, an HTTP GET request will be executed. The *ProcedureType* allows the designer to define other methods as answers. Thus, the user is able to handle a diverse number of these types, for instance, defining URLs for sending notifications via some local URL or a cloud-based URL, depending on the urgency of the action; sending an email, etc.

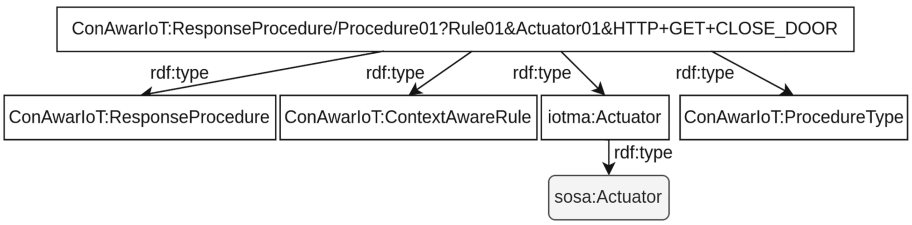


Fig. 4. Example of a ResponseProcedure description

3.3 Relationship with Related Ontologies

ConAwarIoT is based on several existing ontologies, making it compliant with current standards and existing solutions. Most of the concepts are drawn from *SSN/SOSA*, which in turn extends primarily from the *IoTMA* ontology. We also use concepts from *OWL-Time* and *GeoSPARQL* in our ontology.

IoTMA - IoT Monitoring and Actuation ontology [29]**.** IoTMA, which is a previous work of us, is aimed at understanding sensor-actuation contexts, thus incorporating concepts of general monitoring and actuation terms. It incorporates basic semantics for reacting to predefined conditions, in the case of critical events, although, it does not allow defining in a detailed way the conditions for which events to apply and how the system should handle its responses.

Our main contribution in this paper is that of improving the expressiveness of context-awareness rules in IoTMA. This is achieved through the *ContextAware-Rule* concept, which now allows the definition and treatment of more complex and responsive behaviors as required by current IoT applications.

Semantic Sensor Network and Sensor-Observation-Sample-Actuator - SSN/SOSA [15]**.** Provides a lightweight core for defining classes and properties of data managed in the IoT scenario. It supports sensing and actuation device capabilities, for modeling interoperability. Here, we make use of the *sosa:Procedure* for providing the steps for changing the state of the world via an *Actuator*.

GeoSPARQL [21]**.** It is used to describe the location properties of entities. We use *geo:Location* to describe the location of physical entities in the ontology.

OWL-Time [30]**.** It states temporal concepts and properties. *Time* is used to define when there is an *Observation* and an *Actuation* (*time:hasStart*).

3.4 Research Methodology

We have followed the *Design-Science Research* methodology (Hevner et al. [14]): "In this research methodology, a designer answers questions relevant to human problems via the creation of innovative artifacts [...]. The designed artifacts are useful and fundamental to understanding that problems."

At some points of the ontology development, we also follow Noy et al. [20] recommendation: the *Knowledge-Engineering Methodology*. They suggest there is no single way to develop semantics, as domain modeling depends on several factors, including the purpose of the system it supports. The modeling of an ontology is an iterative process: Determine the domain and scope; sketch a list of competency questions that should be answered; consider reusing existing ontologies; enumerate important terms in the ontology; develop the ontology (classes, hierarchy, properties, ...); and, lastly, validate the list of competency questions. In case the outcomes are not as predicted, then return to the first stage.

4 Use Case Description

We focus on a railway use case as an illustration of a cyber-physical system which is very relevant in the IoT scenario. However, it is worth mentioning that the semantics endowed in our ontology are defined at an abstract level and can be applied to different domains, other than railways.

There are several aspects of railway systems that can be improved. One of the duties that must be automated using IoT devices is safety, which involves maintaining the entire railway system. Rail tracks are one of the components in the rail industry that need to be checked on a regular basis. Corrective maintenance is currently carried out based on sparse data and no short-term vision. With this use case, we pave the way for real-time data-based maintenance services, the point at which predictive maintenance begins.

In tracks, certain important elements need to be monitored as they require active maintenance. The geometry of the tracks is an extremely relevant area for ensuring the safe operation of the railway infrastructure. Two of the most important parameters to monitor are the *cant* and *twist* factors of these tracks.

Currently, ADIF[1], the Spanish railway infrastructure manager, uses an auscultation train to check various parameters of the track, including *cant* and *twist* factors. The use of a train to monitor elements on the track has some drawbacks. First, when this train is in use, the normal use of the track is affected, which makes it necessary to take into account in case there is another train with the need to circulate there, and it is usually run in the early hours of the morning. The second is that these checks are run every few months, as due to the great extension of the railway network, this train passes 1 to 2 times a year through each section of the Spanish railway. This implies that if something happens between the checks, no one will be warned about a possible failure.

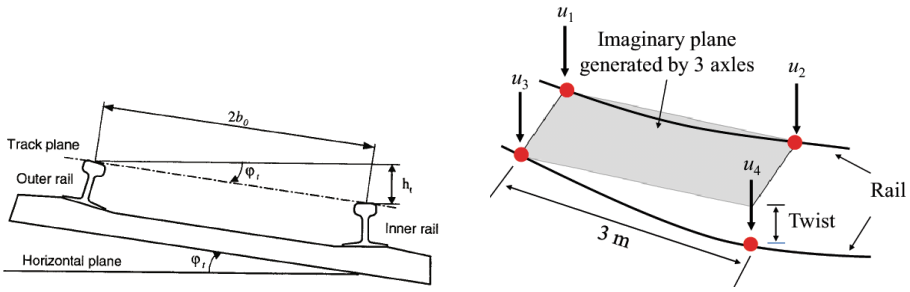
Our method enables the provision of a viable solution to the aforementioned issues, such as continuous monitoring of trains without service interruptions. We consider using IoT devices to perform checks more frequently, and our method is complementary to the train approach. When an anomaly is found, the area becomes of interest, and when work is to be done in a nearby section or underneath the train track, our approach can be employed as a preventive measure.

The first characteristic to be checked, the *cant* factor (Fig. 5a), is the height difference between the two parallel tracks. Using one sensor (tiltmeter[2]), placed in the sleeper, we can provide the difference in angle from one track to the other. The sensor provides information in microvolts for the angle and then, through some calculations, converts it to degrees, and from there, knowing the distance between tracks and basic trigonometry, we can establish the height of the deviation in *millimeters*, as indicated in the ADIF guidelines.

Twist, i.e. the measurement of the rotation of the railway track, (Fig. 5b) is the second feature to monitor. It describes the variation in cross-level measured

[1] https://www.adif.es.

[2] Tiltmeter: A sensitive inclinometer designed to measure very small changes from the horizontal plane, either on the ground or in structures.



(a) Cant (Pombo et al. [25]) (b) Twist (Lee et al. [22])

Fig. 5. Usecase Ilustrations

across five sleepers along the track. This permits comparing the height between each of the four points, resulting in a plane. It can be seen as two *cant* factors at once. In this case, the use of at least two sensors (tiltmeters) is mandatory.

One of the causes of the *cant* factor is the temperature variation in tracks. Tracks expand and contract according to temperature. To correlate this, a sensor near the tracks is also added. Thus, two different sensors need to be monitored at the same time. This is a derived factor from the other already existing one, also contemplated in our use case.

There are several *cant* and *twist* values to compare in the *ADIF* guidelines; based on the measured value, a low-severity warning can be sent, while others can result in high-severity alerts requiring quick action. These elements are monitored in this use case to actuate when certain circumstances are reached.

4.1 Using Our Context-Aware Responsive Ontology

Our solution enables the definition of rules to keep track of the measurements reported by the sensors. Rules can be understood as something simple, like monitoring the events for one sensor, or something more sophisticated, as a combination of different *EventRules* or chained *ConditionRules*. We refer to the former as *simple* and the latter as *complex* to distinguish between them.

Supporting Single Events

In our system, measurements can be compared with other sensor measurements or constants using *ContextAwareRule*s. A case of a simple rule is that of the *cant* factor. It is a feature that can be monitored using a single tiltmeter sensor. This feature consists of using the last measurement to determine the state, and each measurement provides one degree of inclination per axis. For this purpose, Fig. 6 shows the visual representation of a sensor named `TILTMETER_1` that is compared to `COMPARATOR_MORE_THAN` against a constant `CONSTANT_1`.

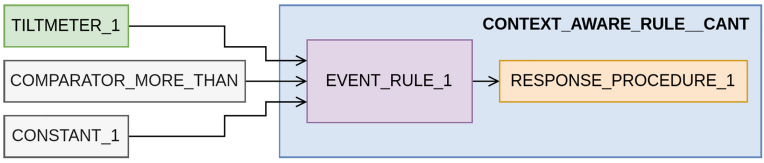


Fig. 6. Representation of a ContextAwareRule with a simple use case - Cant

This evaluation is done in the `EVENT_RULE_1` component, and if it is evaluated as true, it will trigger the `RESPONSE_PROCEDURE_1`.

Supporting Multiple Events or Conditions

More complex rules are useful to model when an element to be monitored depends on two or more variables, such as the *twist* factor, which must be monitored with at least two sensors. One way to measure this factor is to use a pair of tiltmeters and compare the measured values to determine whether each one has a different rotation.

In our system, sensors report measurements from entities to be monitored. If these values are compared with other sensor measurements, constants, or chain conditions, they are called *complex* rules. Figure 7 shows the visual representation of a sensor named `TILTMETER_1` compared to `COMPARATOR_MORE_THAN` against a constant `CONSTANT_1`, evaluated in the `EVENT_RULE_1` component. Additionally, a sensor named `TILTMETER_2` that is compared with `COMPARATOR_MORE_THAN` against a constant `CONSTANT_1`, evaluated in the `EVENT_RULE_2` component.

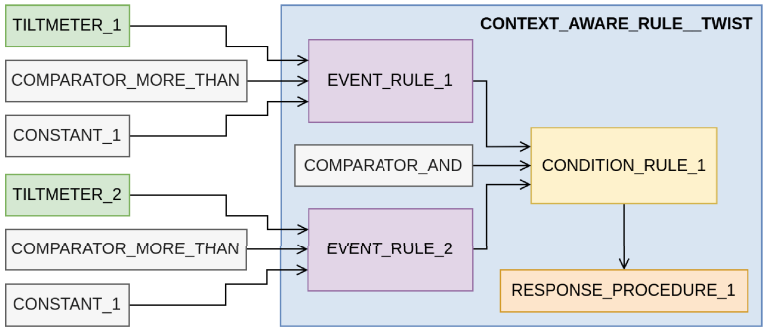


Fig. 7. Representation of a ContextAwareRule with a complex use case - Twist

Furthermore, the two *EventRule* are compared using a `COMPARATOR_AND`, so if both are evaluated as true, it will trigger the `RESPONSE_PROCEDURE_1`.

Another complex example for our use case is to check the correlation between the *cant* or *twist* factor and the temperature. For this, sensor values are compared with other measurements or constants, using the combination of *EventRules* and *ConditionRules*. Figure 8 shows the visual representation of a

sensor named TILTMETER_1 compared to COMPARATOR_MORE_THAN against a constant CONSTANT_1, evaluated in EVENT_RULE_1. Furthermore, a sensor named TILTMETER_2 compared to COMPARATOR_MORE_THAN against a constant CONSTANT_1, evaluated in EVENT_RULE_2. Additionally, a sensor named TEMPERATURE_1 that is compared with COMPARATOR_LESS_THAN against a constant CONSTANT_2, evaluated in the EVENT_RULE_3. Then, EVENT_RULE_1 and EVENT_RULE_2 are evaluated in CONDITION_RULE_1 using a COMPARATOR_OR, and this condition is analyzed together with EVENT_RULE_3 in CONDITION_RULE_2. If both are evaluated as true (COMPARATOR_AND), the RESPONSE_PROCEDURE_1 will be triggered.

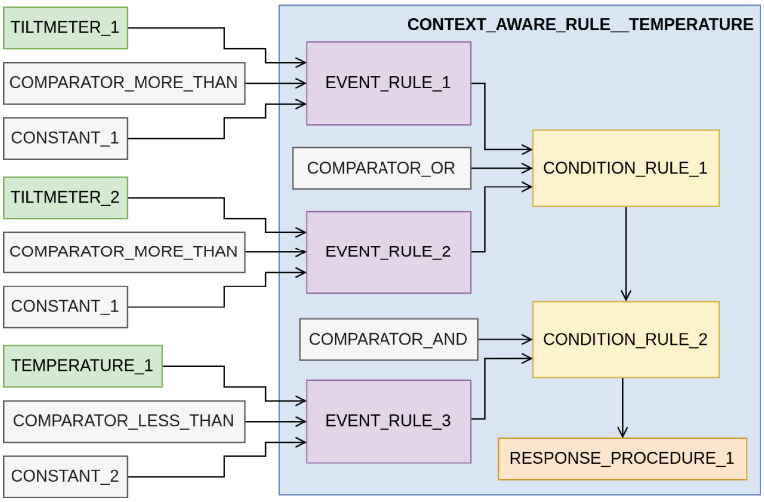


Fig. 8. Represent. of a ContextAwareRule with a complex use case - Temperature

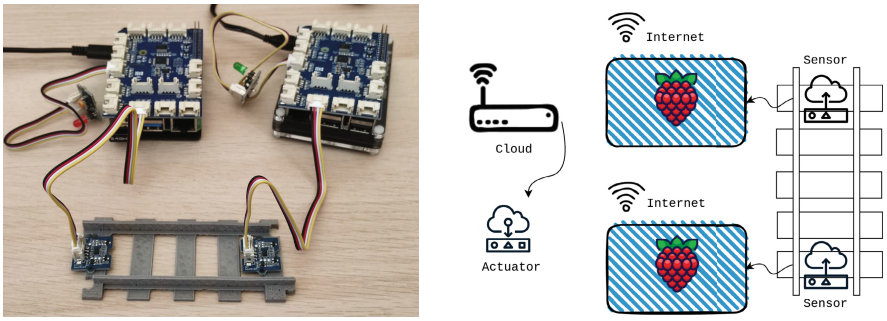
5 Experimentation

We have performed some experiments to demonstrate the validity of our ontology for incorporating context-aware entities to increase semantic interoperability in the IoT. With this purpose in mind, we have established the foundation for our studies on the interoperability of IoT devices in railways. Using the entities provided in our ontology, we handle data interoperability for IoT devices, from monitoring to actuating situations. Although our ontology is sufficiently generic to be used in different domains, the experiment is focused on a specific use case.

Our experimentation is shown in Fig. 9 and aims to demonstrate the *cant* use case. In this way, IoT sensors read information about railway tracks, monitor it, and trigger emails as *ResponseProcedure* if *ContextAwareRule*s are triggered. With this, we enable another way to monitor safety on railways.

Our setup consists of two Raspberry Pi 4B devices to monitor the railway tracks. Each Raspberry reads one track tilt using a tilt sensor as a critical sensor. On top of both Raspberry, there is a GrovePi Shield[3], used to wire sensors to the device. Both devices are capable of reaching the Internet using WiFi mechanisms and submitting measurements to the *Cloud* server, using an HTTP API as the communication method.

In the experiment, each Raspberry is taking measures (Observations) of the track tilt and sending them to the server. We have defined entities to monitor in the *Cloud* server for this purpose. At least two *Things*, one per Raspberry, with two *Sensors*, one per tilt sensor. Also, the *Actuator*, with its module to send emails, has been specified. In addition, the *ContextAwareRules* with its corresponding *EventRules* and *ResponseProcedures* have also been set. The defined *Procedure* is to communicate with an *Actuator* to send an email as a warning.



(a) Setup in the Lab (b) Conceptual View

Fig. 9. Experimentation setup for showcasing the twist scenario

We have developed a context-aware engine in addition to the code that supports the ontology. This engine periodically queries the state of the system in terms of *ContextAwareRule*s. If there are some that have the *executing* flag active, the engine checks the *EventRule*s it has, and together with the *ConditionRule*s, it identifies whether any *ResponseProcedure* has to be performed. If this is the case, it sends it to execution.

In Fig. 10 we see the *Observations* received from a *Sensor* named *TiltSensor*. The tilt sensor of the Raspberry is located on the train track, and the spikes that are seen in the graph represent movements in height of the track, which could be understood as the cant factor value in millimeters when the train runs.

[3] GrovePi Shield for Raspberry Pi: https://www.seeedstudio.com/GrovePi.html.

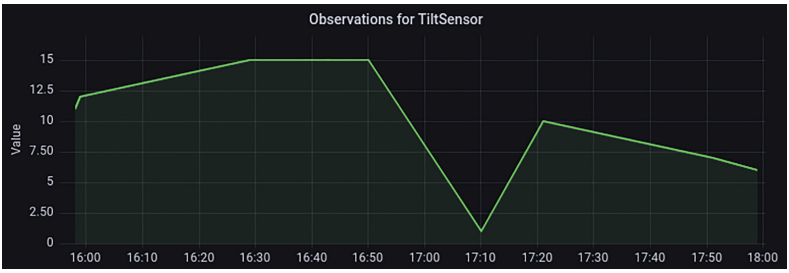


Fig. 10. Our Grafana frontend showing the Observations of the *TiltSensor*

For the graph, we have used Grafana, an open-source data visualization tool. In Grafana we performed an SQL query to the database, where all the experimental information is stored. The *observation* table is where all the metadata information about the *Observation*s is stored, and *obs_integer* to store *Observations* that are *integers*.

```
SELECT observation.time_start as"time",
   obs_integer.value as "value"
FROM observation
INNER JOIN obs_integer ON obs_integer.ID=observation.ID
WHERE observation.sensor_name = 'TiltSensor'
```

In this work, we include a functional code of the ontology, available in https://github.com/worldsensing/conawariot-modelling-context-aware. This code is composed of a backend made in Python Flask and PostgreSQL together with Grafana for visualization. In addition, there is also a first implementation of the context-aware client for the rules, which this ontology supports. As well as the code that supports sensing the *Observations* from the Raspberry Pi.

Our Cloud setup consists of a GCP E2-small instance that has 1 vCore and 2 GB of RAM under Debian 11 Linux. In this instance, we have deployed our own server code. The setup handles the incoming measurements and the context-aware rules. It also maintains the state of the entities using our ontology.

Lessons Learnt: The experiments we have performed in this paper have allowed us to learn some lessons from the approach we have proposed in this paper. They are the following:

- We have been able to show the feasibility of our approach in the sense that we have declaratively and semantically handled the responses to the critical events related to the real-time measurement of the cant and twist factors.
- With our ontology, we have been able to abstract from the technological aspects of IoT devices, thus being able to concentrate on the specification of the responses to the context-aware events.
- The software solution we have implemented in our experiments is able to provide the response to our particular scenario, but it could be easily applied,

without having to change any single line of code, to other railway system characterizations, provided that their IoT infrastructure is specified in terms of our *ConAwarIoT* ontology.

6 Conclusions and Future Work

With a large number of IoT devices being used, the need to homogenize the information that is being manipulated is acknowledged. This work contributes to this homogenization by proposing an ontology for context awareness in the sensing and actuation domains, using IoT devices. The proposed ontology builds upon concepts of well-known ontologies and specifies the knowledge required by context-aware rules for defining the possible reactions to some events if they occur thus enabling the users to define in detail the events needed to be monitored and the actions that modify real-world elements when the conditions are met.

With the proposed railway use case, we enable the possibility of manipulating data in real time in settings where sensors and their potential reactions must be monitored. This use case has been demonstrated using an experiment to showcase the functionality enabled by the proposed ontology. With that, a ready-to-use system is provided, including the backend orchestrated components. In addition, a client code is provided that handles the observations, as well as the context-aware engine code the status and actuations to be executed by *Actuators*.

In future work, we plan to develop a framework that combines the work presented in this article with actual *Worldsensing* IoT devices that use LoRaWAN or related technologies like NB-IoT. Also, other than sending emails when an actuation is desired, introducing more options such as MQTT, emails, and so forth. Further work can go down the Machine Learning path to perform predictive maintenance using the real-time data that is being sent.

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 thank Ignasi Garcia-Milà for his help in the definition of the use case and to the anonymous reviewers for their valuable comments.

References

1. Abowd, G.D., Dey, A.K., Brown, P.J., Davies, N., Smith, M., Steggles, P.: Towards a better understanding of context and context-awareness. In: Gellersen, H.-W. (ed.) HUC 1999. LNCS, vol. 1707, pp. 304–307. Springer, Heidelberg (1999). https://doi.org/10.1007/3-540-48157-5_29
2. Aguilar, J., Jerez, M., Rodríguez, T.: Cameonto: context awareness meta ontology modeling. Appl. Comput. Inform. **14**(2), 202–213 (2018)
3. Alirezaie, M., Renoux, J., et al.: An ontology-based context-aware system for smart homes: E-care@home. Sensors **17**(7), 1586 (2017)
4. Angsuchotmetee, C., Chbeir, R., Cardinale, Y.: MSSN-Onto: an ontology-based approach for flexible event processing in multimedia sensor networks. Futur. Gener. Comput. Syst. **108**, 1140–1158 (2020)

5. Ashton, K.: That internet of things thing. RFID J. **22**(7), 97–114 (2009)
6. Choi, C., Esposito, C., et al.: Intelligent power equipment management based on distributed context-aware inference in smart cities. IEEE Commun. Mag. **56**(7), 212–217 (2018)
7. Dobrescu, R., Merezeanu, D., Mocanu, S.: Context-aware control and monitoring system with IoT and cloud support. Comput. Electron. Agric. **160**, 91–99 (2019)
8. Dörndorfer, J., Hopfensperger, F., Seel, C.: The SenSoMod-modeler - a model-driven architecture approach for mobile context-aware business applications. In: Cappiello, C., Ruiz, M. (eds.) CAiSE 2019, pp. 75–86. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-21297-1_7
9. Elkhodr, M., Shahrestani, S., Cheung, H.: The internet of things: new interoperability, management and security challenges. Int. J. Netw. Secur. Appl. **8**(2), 85–102 (2016)
10. Gaur, S., Almeida, L., et al.: CAP: context-aware programming for cyber physical systems. In: 24th IEEE International Conference on Emerging Technologies and Factory Automation, pp. 1009–1016. ETFA (2019)
11. Gubbi, J., Buyya, R., et al.: Internet of things (IoT): a vision architectural elements and future directions. Future Gener. Comput. Syst. **29**(7), 1645–1660 (2013)
12. Haller, A., et al.: The modular SSN ontology: a joint W3C and OGC standard specifying the semantics of sensors, sampling, and actuation. Semant. Web (2018)
13. Henricksen, K., Indulska, J.: Modelling and using imperfect context information. In: IEEE Annual Conference on Pervasive Computing and Communications Workshops, pp. 33–37. PerCom (2004)
14. Hevner, A., Chatterjee, S.: Design Research in Information Systems: Theory and Practice. Springer, New York (2010). https://doi.org/10.1007/978-1-4419-5653-8
15. Janowicz, K., Haller, A., et al.: SOSA: a lightweight ontology for sensors, observations, samples, and actuators. J. Web Semant. **56**, 1–10 (2019)
16. Jiang, S., Angarita, R., Chiky, R., Cormier, S., Rousseaux, F.: Towards the integration of agricultural data from heterogeneous sources: perspectives for the French agricultural context using semantic technologies. In: Dupuy-Chessa, S., Proper, H.A. (eds.) CAiSE 2020. LNBIP, vol. 382, pp. 89–94. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-49165-9_8
17. Kim, G., Kang, S., et al.: An MQTT-based context-aware autonomous system in oneM2M architecture. IEEE Internet Things J. **6**(5), 8519–8528 (2019)
18. Maarala, A.I., Su, X., Riekki, J.: Semantic reasoning for context-aware internet of things applications. IEEE Internet Things J. **4**(2), 461–473 (2017)
19. Noura, M., Atiquzzaman, M., et al.: Interoperability in internet of things: taxonomies and open challenges. Mob. Netw. Appl. **24**, 796–809 (2019)
20. Noy, N.F., McGuiness, D.L.: Ontology development 101: a guide to creating your first ontology. Technical report, Knowledge Systems - Stanford University (2001)
21. OGC - GeoSPARQL: A Geographic Query Language for RDF Data (2012). https://www.ogc.org/standards/geosparql. Accessed 02 Nov 2022
22. Park, J.W., Lee, K.C., et al.: Traffic safety evaluation for railway bridges using expanded multisensor data fusion. Comput.-Aided Civil Infrastruct. Eng. **31**(10), 749–760 (2016)
23. Perera, C., Zaslavsky, A., et al.: CA4IOT: context awareness for internet of things. In: IEEE International Conference on Green Computing and Communications, pp. 775–782. GreenCom (2012)
24. Perera, C., Zaslavsky, A., et al.: Context aware computing for the internet of things: a survey. IEEE Commun. Surv. Tutor. **16**(1), 414–454 (2014)

25. Pombo, J., Ambrósio, J.: General spatial curve joint for rail guided vehicles: kinematics and dynamics. Multibody Syst. Dyn. **9**, 237–264 (2003)
26. Razzaque, M.A., Milojevic-Jevric, M., et al.: Middleware for internet of things: a survey. IEEE Internet Things J. **3**(1), 70–95 (2016)
27. Schilit, B., Theimer, M.: Disseminating active map information to mobile hosts. IEEE Network **8**(5), 22–32 (1994)
28. Sheng, Q., Benatallah, B.: ContextUML: a UML-based modeling language for model-driven development of context-aware web services. In: International Conference on Mobile Business, pp. 206–212. ICMB (2005)
29. Vila, M., Casamayor, V., Dustdar, S., Teniente, E.: Edge-to-cloud sensing and actuation semantics in the industrial internet of things. Pervasive Mob. Comput. **87**, 101699 (2022)
30. W3C - OWL-Time: Time Ontology in OWL (2020). https://www.w3.org/TR/owl-time/. Accessed 02 July 2022
31. Wang, X., Zhang, D., et al.: Ontology based context modeling and reasoning using OWL. In: IEEE Annual Conference on Pervasive Computing and Communications Workshops, pp. 18–22. PerCom (2004)
32. Xue, L., Liu, Y., et al.: An ontology based scheme for sensor description in context awareness system. In: IEEE International Conference on Information and Automation, pp. 817–820. ICIA (2015)
33. Zhang, Z., Liu, C., Li, X., Han, Y.: A service-based declarative approach for capturing events from multiple sensor streams. In: Pahl, C., Vukovic, M., Yin, J., Yu, Q. (eds.) ICSOC 2018. LNCS, vol. 11236, pp. 255–263. Springer, Cham (2018). https://doi.org/10.1007/978-3-030-03596-9_17