Making heterogeneous smart home data interoperable with the SAREF ontology

Van Der Weerdt, Roderick; De Boer, Victor; Daniele, Laura; Nouwt, Barry; Siebes, Ronald

published in
International Journal of Metadata, Semantics and Ontologies
2022

DOI (link to publisher)
10.1504/ijmso.2021.125893

document version
Publisher's PDF, also known as Version of record

Making heterogeneous smart home data interoperable with the SAREF ontology

Roderick Van Der Weerdt* and Victor De Boer

Vrije Universiteit Amsterdam,
Amsterdam, North Holland, The Netherlands
Email: r.p.vander.weerdt@vu.nl
Email: v.de.boer@vu.nl
*Corresponding author

Laura Daniele and Barry Nouwt

TNO – Netherlands Organisation for Applied Scientific Research,
The Hague, The Netherlands
Email: laura.daniele@tno.nl
Email: barry.nouwt@tno.nl

Ronald Siebes

Vrije Universiteit Amsterdam,
Amsterdam, The Netherlands
Email: r.m.siebes@vu.nl

Abstract: SAREF is an ontology created to enable interoperability between smart devices, but there is a lack in the literature of practical examples to implement SAREF in real applications. We validate the practical implementation of SAREF through two approaches. We first examine two methods to map the IoT data available in a smart home into linked data using SAREF: (1) by creating a template-based mapping to describe how SAREF can be used and (2) by using a mapping language to demonstrate it can be simple to map, while still using SAREF. The second approach demonstrates the communication capabilities of IoT devices when they share knowledge represented using SAREF and describes how SAREF enables interoperability between different devices. The two approaches demonstrate that all the information from various data sets of smart devices can successfully be transformed into the SAREF ontology and how SAREF can be applied in a concrete interoperability framework.

Keywords: IoT; SAREF; ontology; data mapping; smart home.

Reference to this paper should be made as follows: Van Der Weerdt, R., De Boer, V., Daniele, L., Nouwt, B. and Siebes, R. (2021) 'Making heterogeneous smart home data interoperable with the SAREF ontology', *Int. J. Metadata Semantics and Ontologies*, Vol. 15, No. 4, pp.280–293.

Biographical notes: Roderick Van Der Weerdt is a PhD student in the User-Centric Data Science research group at Vrije Universiteit Amsterdam. He graduated in Artificial Intelligence from the University of Amsterdam. Currently he works on the combination of Machine Learning with Knowledge Graphs focussing on Knowledge graphs of IoT data.

Victor De Boer is an Associate Professor of User-Centric Data Science at Vrije Universiteit Amsterdam. He is also a senior research fellow at Netherlands Institute for Sound and Vision and co-director of the Cultural AI Lab. He works on data integration, semantic data enrichment and knowledge sharing using Linked Data technologies in various domains including Cultural Heritage, Digital Humanities and ICT for Development.

Laura Daniele is a senior scientist in the area of ontology engineering and semantic technologies at TNO. She is the creator of the SAREF ontology for the European Commission, standardised by ETSI, and serves as expert in ETSI SmartM2M Technical Committee. She is also active in AIOTI, where she is the co-leader of the Semantic Interoperability group of WG03 on IoT Standardisation. She is currently lead scientist in the H2020 Interconnect large scale pilot, which aims at providing interoperable solutions connecting smart homes, buildings and grids.

Barry Nouwt (MSc) is a medior Scientist semantic technology at TNO within the Data Science department. Until 2015, he worked with SemLab B.V. on commercial applications of NLP and semantics, primarily in the Financial and Government domain. At TNO, his research activities centre around ontologies, model-driven development and semantic reasoning with a focus on

increasing the value of formalised domain knowledge. He applies his research results and software engineering skills to diverse projects and domains.

Ronald Siebes received his PhD in Artificial Intelligence from the Knowledge Representation and Reasoning group at the VU University Amsterdam, The Netherlands. Currently he works as a senior researcher at the same university in various national and European projects involving Semantic Web technology applied to Social Sciences, Humanities and the IoT.

This article is a revised and extended version of the paper entitled 'Validating SAREF in a Smart Home Environment' presented at the '14th International Conference on Metadata and Semantics Research, MTSR 2020', Madrid, Spain, 2–4 December 2020.

1 Introduction

Over the last years a trend can be observed where many traditional consumer products get 'smart' capabilities and get connected to the Internet of Things (IoT). Connecting multiple devices brings the promise that all those devices are now able to interact with each other in meaningful ways. The reality is that devices are not nearly as interconnected as they can be (Hsu and Lin, 2016), often devices can only be accessed from specific (vendor-based) apps, other times the devices are not able to communicate because they do not speak the same language. In order to make the interoperability between smart devices within IoT possible the Smart Applications REFerence ontology (SAREF) has been created (Daniele et al., 2015), an ontology specifically designed to encompass the information that smart devices need to exchange in order to have meaningful interactions.

SAREF has been validated before (Bajaj et al., Moreira et al., 2017) to determine the quality of the ontology. But until now there is very limited research on applying the SAREF ontologies, validating if it satisfies the requirements of mapping realistic data from physical devices.

The goal of this paper is to create examples of a practical implementation, validating the ontology in two ways, its ability to express all the information available from multiple data sets with data from smart devices in a home and its ability to enable interoperability by representing messages in a meaningful manner allowing for communication between smart devices.

To achieve the first part we map data from two different data sets with data from multiple smart homes into Linked Data using SAREF (see Section 3). The mapping of the first data set is performed using a simple template to be able to describe every step of mapping a data set and how to use SAREF in that mapping. The second mapping is performed with the YARRRML mapping language, which allows for an easier mapping process and demonstrates that SAREF is usable with existing mapping languages.

For the second part, we implement a knowledge exchange framework to validate the effectiveness of using SAREF for communication (see Section 4). The experiment is performed in two steps, the first step describes how the SAREF ontology can be used to express the information that is exchanged. The second step introduces more information that originated from different data sets, demonstrating how SAREF is used to make the devices that use information from both the data sets interoperable.

2 Related work

To provide a background to our experiments, we will first present an in-depth examination of the SAREF ontology and its extensions, followed by three related ontologies and other validation studies using competency questions to validate ontologies. We finally describe the RML mapping language and how it can be used to map data available in different formats to a knowledge graph.

2.1 SAREF

The Smart Applications REFerence ontology (SAREF) is a reference ontology for IoT applications (ETSI TS, 2020). Its development started in 2013, when the European Commission, in collaboration with the European Telecommunication Standardisation Institute (ETSI), launched an initiative to build a common ontology in close collaboration with the smart appliances industry (Daniele et al., 2015). In a fragmented landscape of IoT standards, platforms and technologies across different vertical domains (ETSI TS, 2016a, 2016b), SAREF was created as a shared model of consensus that could enable the communication among various IoT devices from different manufacturers that use different protocols and standards (Daniele et al., 2018). The first proof-of-concept solution based on SAREF was demonstrated and implemented in 2017 on existing commercial products in the energy domain (Moreira et al., 2017). SAREF is published as a series of ETSI technical specifications, consisting of a modular framework that comprises a generic core ontology for loT (ETSI TS, 2020) and 11 domain-specific extensions (ETSI TS 103 410, parts 1–11), such as SAREF for Energy, Buildings and Cities, amongst others. The SAREF framework is maintained and evolved by ETSI and experts from several European organisations that successfully collaborate with each other. One of the latest supported initiatives is the development of an open portal for the SAREF community and industry stakeholders, so that they can contribute directly to the SAREF evolution (https://saref.etsi.org/).

Figure 1 shows an overview of the main classes and relationships of the SAREF core ontology. In total SAREF contains 81 classes, 35 object properties and 5 data properties. The starting point is the concept of *Device*, which is defined as a tangible object designed to accomplish a particular *Task*. In order to accomplish this task, the device performs a

Function. For example, a temperature sensor is a device of type `saref:Sensor` is designed for tasks such as `saref:Comfort`, `saref:WellBeing` or `saref:EnergyEfficiency`, and performs a `saref:SensingFunction`. Functions have commands. A *Command* is a directive that a device needs to support to perform a certain function. Depending on the function(s) it performs, a device can be found in a corresponding *State*. A device that wants (a certain set of) its function(s) to be discoverable, registerable and remotely controllable by other devices in the network can expose these functions as a *Service*. A device can also have a *Profile*, which is a specification to collect information about a certain *Property* or *Commodity* (e.g. Energy or Water) for optimising their usage in the home/building in which the device is located. A *Property* is defined as anything that can be sensed, measured or controlled by a device and is associated to measurements. For example, a temperature sensor measures a property of type `saref:Temperature`. A *Measurement* contains the measured value made over a property and must be associated to an unit of measure and a timestamp. The *Feature of Interest* concept further allows to represent the context of a measurement, i.e., any real world entity from which a property is measured. For example, whether the measured temperature is that of a room or of a person. A more detailed description of the SAREF classes and properties can be found in ETSI TS (2020).

2.2 Related ontologies

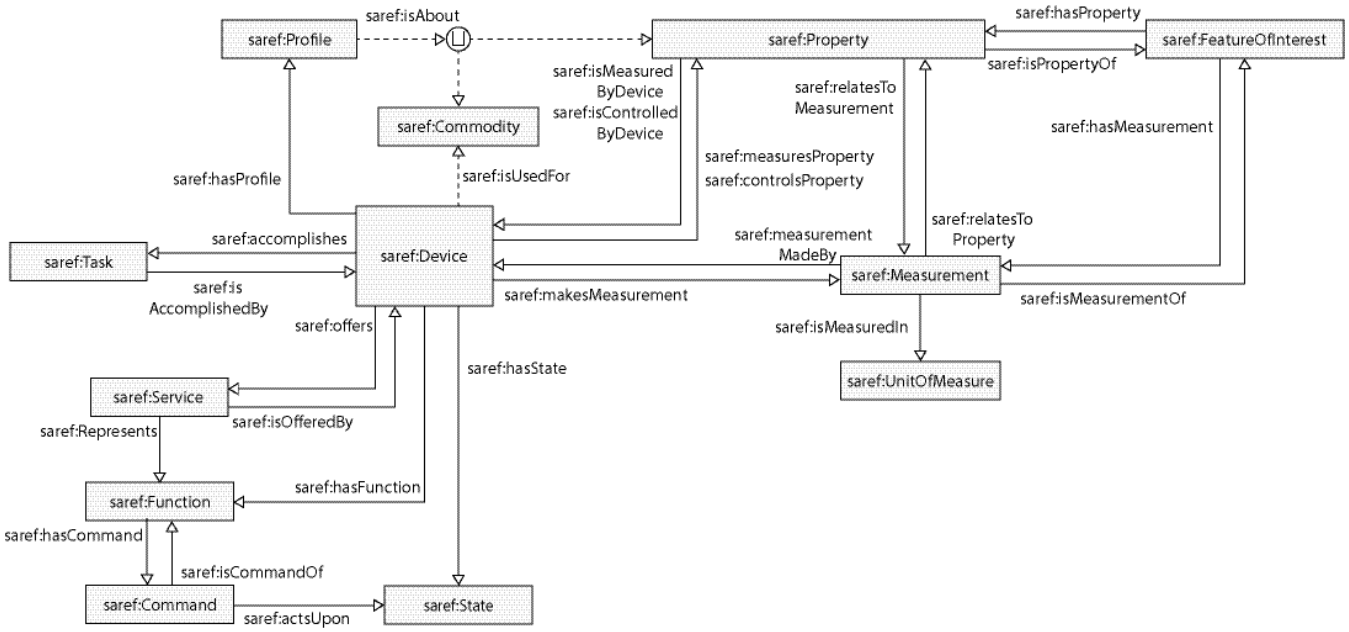
The Semantic Sensor Network (SSN) ontology, was specifically created for the modelling of sensors and their observations (Compton et al., 2012). This strict focus on sensors allows for a modular use of the ontology with other ontologies. A sensor is defined as 'anything that observes', which means it does not include other types of smart devices like locks and lights. The Sensor, Observation, Sample and Actuator (SOSA) ontology is an extension and partial replacement of SSN created to model 'the interaction between the entities involved in the acts of observation, actuation and sampling' (Janowicz et al., 2019). Because SSN was considered too rigid SOSA was created to be more flexible, having 13 concepts and 21 properties compared to the 41 concepts and 39 object properties of SSN.

The Web of Things (WoT) Thing Description (TD) is a more recent ontology that was also created by W3C (2020). This ontology models the metadata of the devices instead of the measurements. The creators envision that every smart device will have its own TD to serve as identification and as a starting point for interaction with the device, similar to an index page of a website. However, although complementary, it has insufficient detail to cover the full information provided in the measurement data from the IoT devices in our evaluation.

2.3 Competency questions for validation

Validation of an ontology can be done by creating a scenario and formulating a set of competency questions. Competency questions are proposed as a way to informally define requirements for an ontology, the resulting ontology should be able to answer all of the competency questions to prove it is the correct ontology to use in this scenario (Gruninger and Fox, 1995).

Figure 1 Overview of the SAREF ontology[1]



Competency questions have been used to validate SAREF before, as demonstrated in the research by Moreira et al. (2017). In their work they created a model that maps a knowledge graph using the SSN ontology into a knowledge graph using the SAREF ontology. Using an example situation of a wind sensor represented using the SSN ontology they were able to map the data to a knowledge graph, but were not able to keep all of the information that was available in the original representation, hence resulting in unwanted information loss. Competency questions were used to define the requirements that the ontologies required and showed that the new knowledge graph did not retain enough information from the original. The experiment they performed used an example specifically created for SSN, whilst in this paper we will use data that comes from smart devices in a home or office setting, which is what SAREF was developed for and should therefore allow for a better evaluation of SAREF.

Another example of competency questions being used as a tool to select different ontologies can be found in Bajaj et al. (2017). Bajaj et al. (2017) used the competency questions they define to determine which classes an ontology requires to fit their requirements. For example, SAREF was rejected by the authors as a suitable ontology because it lacked a class to define the concept of location. Since then a new addition to the SAREF ontology has been the `saref:FeatureOfInterest` class, which allows for exactly such a definition.

Sagar et al. (2018) extended the SOSA/SSN ontology in order to allow for smart sensors, sensors that are able to select and run different algorithms resulting in different data, to be modelled accurately. They created 13 competency questions and based on those reject the existing ontologies, including SAREF. The S3N ontology that they propose was able to resolve all the competency questions. Instead of trying to model the algorithms behind 'smarter' observations, the decision was made to only look at the output of the devices with our implementations since this is the relevant information that we have available. Similarly, we only look at the output of a thermometer and not at the intricate ways the device measures the temperature of the room.

2.4 Mapping languages

The mapping from existing data sets to Linked Data can be performed with a template, by writing the data triples by hand and writing a script to fill in the variables, or procedurally, by creating a ruleset that a programmed interpreter can follow to create the Linked Data from the data set. Linked Data uses the RDF data model (Lassila et al., 1998) to represent data. It uses RDF triples (subject, predicate, object) as its main data model. Existing data (e.g. CSV or XML form) therefore needs to be converted to these RDF triples. RML is an RDF Mapping Language created as a general mapping language (Dimou et al., 2014), meaning it can be used to map data sources with different data formats (XML, CSV, etc.) using the same rules. The rules iterate over the data, e.g. with CSV structured data it iterates over the rows or with json structured data it iterates over the objects, and create new Linked Data triples following the rules defined in the mapping ruleset using information from the data sets.

The rules were primarily developed to be machine-processable and can become convoluted and more difficult to interpret by humans. YARRRML (Heyvaert et al., 2018) developed to counter this issue. It added an extra step to the mapping process by first allowing the user to write the rules in a more human-friendly style which are then converted into regular RML, sparing the user from the more difficult to interpret RML language. The ruleset is made up of a mapping for each subject, to create triples of shape $<$subject, property, object$>$. Each mapping can create multiple triples based on all the property object pairs that are included in the mapping.

We use YARRRML to create the rules for the mapping of our second data set. The following section goes into detail on how we used it.

3 Mapping data sources

The main purpose of SAREF is to enable interoperability between different devices and services (ETSI TS, 2020). In order to validate SAREF, the mapping experiments presented in this section will focus on whether SAREF is capable of expressing all of the information from a smart home by mapping the smart devices to a knowledge graph using SAREF.

The two approaches to map existing data to SAREF are different in multiple ways, those are listed here. The template based mapping allows for simple writing of reusable templates. These templates take the form of SAREF-compliant Turtle (an RDF file format), with placeholder elements, to be filled in with values from the existing date. A benefit of this approach is that a user familiar with SAREF and Turtle can relatively easily write such templates. Using custom Python functions, values can be combined or manipulated to match the desired output. However, with more complex data, this approach does not scale well, and there is a danger that many custom functions make the mapping approach difficult to reuse. Especially in a scenario where multiple users with existing data are expected to write their own mappings, such reusability is required. Therefore, the mapping-language approach using YARRRML allows for the scalable creation of explicit, declarative and interpretable rules. Here, too, we can base the rules on the desired outcome in SAREF compliant Turtle, but the increased expressivity of YARRRML allows for declarative expression of the more complex mapping rules. In a scenario where multiple parties exchange heterogeneous data, such a rule language is therefore key. An additional benefit is that the mapping language works on input data of multiple types (CSV, XML, etc.).

3.1 Mapping with a template: synthetic data set

The first mapping is performed with synthetic, yet realistic data from 33 smart devices in a smart home that transforms the data into a knowledge graph using the SAREF core ontology. We use synthetic data here since this is the most diverse set of devices from one smart home that we could find. Data sets either only supply one sort of data, such as the OPSD data (which is described in Sub-section 3.3) only supplying energy consumption for different devices, or there is little to no data available in a data set, such as with this data set. Generating synthetic data based on the information we have about the devices enables us to have both, i.e., a data set with data that is also diverse. This synthetic data is generated by interpreting the possible range of values the device is able to supply. For the measurement devices this is based on their unit of measure, such as percentage, Parts Per Million (PPM) or, degrees Celsius. For the sensors this can be a state such as on/off or yes/no. These patterns are provided in the context of a pilot led by a large construction company in the Netherlands experimenting with adding smart devices to the offices and houses they build and, as such, represent realistic smart home data.

For the devices presented in this section we do not differentiate between similar smart devices that are used multiple times in the house. For example, all five rooms (bedroom, kitchen, bathroom, hallway and living room) have a motion detector and presence detector, but except for the feature of interest being a different room, this does not affect the corresponding mapping. Therefore each device is only described once, independently of what room it is located in. This resulted in the following nine data sources that are used for this experiment:

- *current temperature*, measurement of the room temperature in degrees Celsius.
- *CO_2 levels*, measurement of the ppm CO_2 in a room.
- *last time movement detected*, marks the most recent moment (timestamp) that movement was detected in a room. Used to determine when residents are at home.
- *first-time movement detected*, marks the beginning (timestamp) of new movement detected in a room. Used to determine when residents are at home.
- *motion detected*, returns a yes or no state based on whether the sensor detected motion.
- *presence detected*, returns a yes or no state based on whether the sensor detected presence, based on the CO_2 levels in a room.
- *humidity*, measurement of the percentage of humidity in a room.
- *smart switch actuator*, an actuator that can be turned on or off by sending it an on/off command.
- *target temperature*, the desired temperature of a room in degrees Celsius, controlled by the thermostat.

All the devices send updated information every five seconds, which is recorded with a timestamp. The complexity of the data from the devices differs from a basic *thermometer* to a more elaborate *last time movement detected* sensor. As it is discussed in Sub-section 2.3, we only focus on the output of the device and not on the internal calculations.

The combination of these data sources can be used in multiple scenarios. A possible scenario is to adjust the room temperature when people are not present. Using a combination of the *first-time movement detected* data and the *last time movement detected* data, which in turn is based on the *motion detected* and *presence detected* data, we can determine whether a home is occupied. When people are not present the target temperature could be set to a lower setting, meaning lower as opposed to the target temperature when there are people present. When the current temperature is below the target temperature a command can be send to the heating system of the home (in the case that one of the smart switch actuators is connected to a heating system) shutting it down. This would allow a home to only be heated when it is needed.

3.2 Mapping with a template: implementation

Not all classes shown in Figure 1 are used in the mapping for a data source. For example, a command that is sent to a smart actuator will not be using the `saref:Measurement` class. The Technical Specification (ETSI TS, 2020) describing the SAREF ontology is used to make a selection of relevant classes for each smart device. The relevant classes for each device can be seen in Table 1.

Most mappings are easily performed by selecting the straightforward SAREF class, e.g. for the `saref:Property` class in the mapping of the thermometer we can use its subclass `saref:Temperature` to mean that the measurement in this graph pattern relates to temperature.

For the sake of clarity, this section only contains mappings for two data sources from this data set and a summary of the special cases where the decisions made need an explanation. The complete mapping of all the data sources can be found online[2] where all the mappings are represented as triples.

3.2.1 In-depth mapping explanations

The two examples that are chosen to be clarified are the data from the thermometer and the data from the CO_2 sensor. The thermometer mapping is a mapping where all the relevant classes are already available. For the CO_2 sensor two new subclasses have to be added. The OM1.8: Units of Measure ontology (Rijgersberg et al., 2013) was chosen to represent the units of measure, as suggested in ETSI TS (2020). For the humidity sensor we only clarify the `UnitOfMeasure`, since this was the only other device that required an additional subclass to be created.

Table 1 Overview of the classes used to model each data source

	current temperature	*CO2 levels*	*first time movement detected*	*last time movement detected*	*motion detected*	*precense detected*	*humidity*	*smart switch actuator*	*target temperature*
`saref:Measurement`	x	x					x		x
`saref:UnitOfMeasure`	x	x					x		x
`saref:Property`	x	x	x	x	x	x	x	x	x
`saref:FeatureOfInterest`	x	x	x	x	x	x	x	x	x
`saref:Device`	x	x	x	x	x	x	x	x	x
`saref:Commodity`									
`saref:Profile`									
`saref:Task`	x	x	x	x	x	x	x	x	x
`saref:Service`								x	
`saref:Function`	x	x	x	x	x	x	x	x	x
`saref:Command`								x	
`saref:State`			x	x	x	x		x	

3.2.2 Thermometer

The thermometer makes a measurement of the temperature in a room. The mapping requires the following:

- `saref:Measurement` class for the value of the temperature and timestamp.
- `saref:UnitOfMeasure` class for the unit of the measurement. We can use the subclass `saref:TemperatureUnit` and `om:degree_Celsius` as its instance.
- `saref:Property` class to map what the measurement is measuring, in this case temperature, so that we can use its subclass `saref:Temperature`.
- `saref:FeatureOfInterest` class represents the feature of interest, in this case the room of which the measurements are made. The Uniform Resource Identifier (URI), the unique name for specific objects in the Linked Data, would represent the name of the room.
- `saref:Device` class to represent the device itself that is making the measurements, we can use the subclass `saref:Sensor`.
- `saref:Task` class to represent the goal of `saref:Comfort` for the user.
- `saref:Function` class to define the function as a `saref:SensingFunction`.

3.2.3 CO_2 sensor

The CO_2 sensor makes a measurement of the CO_2 levels in a room. The mapping requires the following:

- `saref:Measurement` class for the CO_2 levels and the timestamp.
- `saref:UnitOfMeasure`, SAREF does not have a unit of measure subclass for CO_2, a new subclass of `UnitOfMeasure` is created to represent CO_2 measurements and the parts per million unit of OM1.8 was used as unit instance, resulting in the additional triple:

  `om:parts_per_million rdf:type ex:`CO_2`Unit`
- `saref:Property`, in this case CO_2 levels, for which a class in not available in SAREF, but can be easily added:

  `<PROPERTY_URI> rdf:type ex:`CO_2
- `saref:FeatureOfInterest`, similar to the thermometer mapping.
- `saref:Device`, similar to the thermometer mapping.
- `saref:Task` class to represent the goal of the `saref:WellBeing` of the user.
- `saref:Function`, similar to the thermometer mapping.

3.2.4 Humidity sensor

The humidity sensor data is similar to the CO_2 sensor data where an additional subclass for `saref:UnitOfMeasure` was created to express the humidity measurements:

```
om:percent rdf:type ex:HumidityUnit
```

3.2.5 Mapping results

Using SAREF we are able to represent all of the information available in the data set and additional information we have about the devices, such as the task and function, in Linked Data. The three exceptions are mentioned above but are easily extended.

3.3 Mapping with a mapping language: OPSD household data set

We perform another data mapping, this time with the OPSD household data set, to demonstrate that SAREF also allows mappings to be created with a mapping language. This mapping is done with the OPSD Household Data set (Open Power System Data, 2020), containing data from 68 devices from 11 different buildings, 3 industrial buildings, 2 public buildings and 6 residential households located in Konstanz. For this experiment, we only focus on devices in the six residential households. Similar to the synthetic data set we only describe each device below once, independent of which residence they were located in.

- *grid_import*, Energy imported from the public grid, in kWh.
- *grid_export*, Energy exported to the public grid, in kWh.
- *pv*, Total Photovoltaic energy generation, in kWh.
- *dishwasher*, Dishwasher energy consumption, in kWh.
- *ev*, Electric Vehicle charging energy, in kWh.
- *refrigerator*, Refrigerator energy consumption, in kWh.
- *freezer* Freezer energy consumption, in kWh.
- *washing_machine* Washing machine energy consumption, in kWh.
- *circulation_pump* Circulation pump energy consumption, in kWh.

Multiple variants of the data set were available, with the data being retrieved every single minute, every 15 minutes, or every hour. For our purpose of creating a mapping the time interval was not important, the hour interval was chosen to have a clearer difference between each consequent data value.

As opposed to the synthetic data set, this data set contains five years' worth of data points. Another difference is the fact that the synthetic data set has multiple different devices with multiple different kinds of measurements (e.g. percentage, degrees Celsius) of states (e.g. on/off, yes/no) as opposed to the OPSD households data set, which does have multiple different devices, but all with the same type of measurement in kWh. The relevant SAREF classes are also similar for each device, since they represent the same measurements, as seen in Table 2. This puts more emphasis on the mapping process, since there is less of an automatic distinction between the mappings from the devices.

Table 2 Overview of the classes used to model each device from the OPSD household data set

	grid_import	*grid_export*	*pv*	*dishwasher*	*ev*	*refrigerator*	*freezer*	*washing_machine*	*circulation_pump*
`saref:Measurement`	x	x	x	x	x	x	x	x	x
`saref:UnitOfMeasure`	x	x	x	x	x	x	x	x	x
`saref:Property`	x	x	x	x	x	x	x	x	x
`saref:FeatureOfInterest`	x	x	x	x	x	x	x	x	x
`saref:Device`	x	x	x	x	x	x	x	x	x
`saref:Commodity`									
`saref:Profile`									
`saref:Task`	x	x	x	x	x	x	x	x	x
`saref:Service`									
`saref:Function`	x	x	x	x	x	x	x	x	x
`saref:Command`									
`saref:State`									

3.4 Mapping with a mapping language: implementation

Instead of using a simple template to create a mapping, as it is done with the mapping in Sub-section 3.2, we can use a mapping language to describe with rules how the data must be mapped. RML is a popular and effective language for such rules and YARRRML provides a user-friendly interface for producing such rules, as described in Sub-section 2.4. For the OPSD household data set, we created a YARRRML ruleset for each device in each of the six residences.

3.4.1 Rules explanation

Since each ruleset is about 190 lines long, we will only describe some of the rules in-depth and explain how those generalise to the rest of the rules. The complete rulesets and mappings can be found online.[3]

Listing 1.1: Part of the YARRRML ruleset for the mapping of the OPSD data

```
1 measurement:
2 sources: data 1
3 s: interconnect:measurement_OPSDres4_
WM_$ (utc_timestamp)
4 po:
5 - [a, saref:Measurement]
6 - [saref: hasValue, $
(DE_KN_residential4_washing_machine),
xsd: double]
7 - [saref:isMeasuredIn, om:
kilowatt_hour ~ iri]
8 - [saref:hasTimestamp , $
(utc_timestamp), xsd: dateTime]
9 - p: saref:isMeasurementOf
10 o:
11 mapping: featureOfInterest
12 condition:
13 function: equal
14 parameters:
15 - [ str1, $ (utc_timestamp), s]
16 - [ str2, $ (utc_timestamp), o]
17 ...
```

The ruleset has been set up to create a specific mapping for each class required for the total mapping of the data of each device. Shown in Listing 1.1 is the mapping for the `saref:Measurement` class for the washing machine.

Line 1 is the name of the mapping, which can be used to refer to other mappings within this ruleset.

Line 2 refers to the data file used, YARRRML supports multiple files mapped at the same time, but for this experiment, we only map one file at a time.

Line 3 defines the URI of this class instance, in this case the URI for one specific measurement for residence *OPSDres4* from the washing machine *wm* at timepoint *timestamp*. The linked data created by this ruleset is of shape < subject, property, object > . This URI is the subject (s).

Line 4 starts defining the property-object pairs, made up of: the property (p), the relation between subject and object, and object (o), the target of the relation. Below this line, all the po pairs are defined.

Line 5 defines the class of this instance, in this case, the measurement class.

Line 6 contains the value of the measurement, as taken from the data set, once for each line in the column `DE_KN_residential4_washing_machine`.

Line 7 defines the unit of measurement instance, here from the OM ontology.

Line 8 retrieves the timestamp from the data set, because the data set is processed line by line the timestamp and value (on Line 6) will come from the same line.

Line 9 creates a property that targets another mapping in the ruleset, allowing for mappings to other instances of which the URI is not yet known, in this case to the `FeatureOfInterest` mapping.

Line 10 defines the parameters to make sure that relations are only created between instances that are supposed to be connected.

For each class used in the mapping of the devices (see Table 2) a mapping is created following this same structure, with all the relations between the classes following the structure in Figure 1.

3.4.2 In-depth mapping explanations

Since the only mapping differences are already explained in Table 3, there is no need to describe all the mappings individually. Below follows the explanation of the mapping of the washing machine as an example.

3.4.3 Washing machine

The Washing machine provides its total aggregated power consumption. The mapping requires the:

- `saref:Measurement` class for the value of the power consumption and timestamp.
- `saref:UnitOfMeasure` class for the unit of the measurement, we can use the subclass `saref:PowerUnit` and `om:kilowatt_hour` as its instance.
- `saref:Property` class to map what the measurement is measuring, in this case power consumption, so we can use its subclass `saref:Power`.
- `saref:FeatureOfInterest` class represents the feature of interest, in this the residence of which the measurements are made. The URI represents the name of the residence.
- `saref:Device` class to represent the device itself that is making the measurements, we can use the subclass `saref:Appliance`.
- `saref:Task` class to represent the goal of `saref:Washing` for the user.
- `saref:Function` class to define the function as a `saref:MeteringFunction`.

Table 3 Overview of the subclasses chosen to represent the devices from the OPSD household data set in the mapping

Device name	*saref:Device*	*saref:Task*	*saref:Function*
grid_import	saref:Meter	saref:MeterReading	saref:MeteringFunction
grid_export	saref:Meter	saref:MeterReading	saref:MeteringFunction
pv	saref:Meter	saref:MeterReading	saref:MeteringFunction
dishwasher	saref:Appliance	saref:Washing	saref:MeteringFunction
ev	saref:Meter	saref:MeterReading	saref:MeteringFunction
refrigerator	saref:Appliance	saref:WellBeing	saref:MeteringFunction
freezer	saref:Appliance	saref:WellBeing	saref:MeteringFunction
washing_machine	saref:Appliance	saref:Washing	saref:MeteringFunction
circulation_pump	saref:Device	saref:Comfort	saref:MeteringFunction

3.4.3 Mapping results

All of the available information is easily represented with SAREF as Linked Data. The difficulty is selecting more descriptive task and function subclasses to describe the data of different devices, so their distinctive features are highlighted. Using YARRRML to create the mapping rules in accordance with SAREF is beneficial, it is faster, and easier to alter for additional similar (but not identical) devices.

4 Sharing data with SAREF

This Section describes the experiment to test the capability of SAREF to enable interoperability between different devices. The experiment is presented in two parts, first we focus on the subset of a scenario to demonstrate the core idea of message interaction between different devices. The second part includes the complete scenario with additional devices, demonstrating the capability of SAREF to combine data from different devices and data sets. The next section details the scenario and the competency questions are formulated accordingly. To achieve interoperability, multiple separate devices need to be set up that 1) do not have direct or hard links between them and 2) communicate exclusively using SAREF. To the best of our knowledge, no other ontology-based interoperability framework in smart homes allows for the data to remain at the source where it is produced. Our findings should generalise to other interoperability frameworks. The Knowledge Engine is shortly described in Sub-section 4.2, followed by both experiments and their results.

4.1 Scenario and competency questions

As a scenario we choose a temperature-controlled home, controlled by a thermostat that uses observations about the home, made by different devices, to determine whether the heating device in the home should be turned on or off based on the current temperature of the room and only does so when the home is occupied. The information made available by the devices are: the current temperature of the home, energy consumption of multiple different appliances, target temperature of the home and whether the heating device should be heating the room or not. Based on this scenario we define the following four competency questions:

1 What is the temperature of a specific room?

2 Is this higher or lower than the threshold temperature?

3 Should the heater be turned on or off?

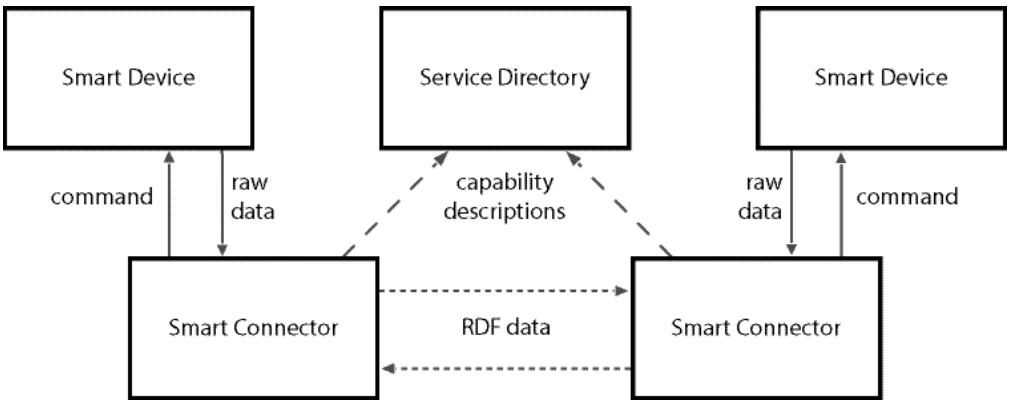
4 Are devices consuming energy that indicate occupancy of the home?

We address CQ-1 to CQ-3 in a first experiment where the design and implementation of the interoperability framework are introduced. These also only use the first data set introduced in Section 3. CQ-4 is addressed by an extended version of the setup and uses both heterogeneous data sets from Section 3. The framework we use to make the message interactions possible is described in the next section.

4.2 Knowledge Engine

The Knowledge Engine is a framework that allows multiple IoT devices to exchange knowledge in an interoperable way within a network. It is used by adding a smart connector to each device and registering this smart connector in the service directory, a visualisation can be found in Figure 2. The smart connector provides two functions to achieve interoperability: translation and discovery. It *translates* between a common ontology and the specific language of the device it is connected to, allowing devices from different vendors to understand each other. It also dynamically *discovers* other smart connectors that can supply relevant data, which prevents hard links between devices. These two functions of the smart connector allow connections between devices in a network to be solely based on concepts and relations from a common ontology. This means that any device in the network can potentially be replaced by a similar one from a different vendor without loss of function, i.e., they are interoperable with each other.

Figure 2 Visualisation of the data flow in the Knowledge Engine



Both the translation and the discovery functions of the smart connector require configuration. Regarding the translation, this configuration consists of custom code provided by a developer that maps the specific device language to the common ontology. For discovery, the smart connector needs to be configured with the capabilities of the device (i.e., the *knowledge demand* and *knowledge supply*). The demand describes (in terms of the common ontology) the data that the device requires to function, while the supply describes the data that the device can provide to other devices. These capability descriptions are similar to the SPARQL query language syntax to describe graph patterns, see Figure 3(a). Apart from this configuration, the discovery function also requires a component called 'Service Directory' to which all smart connectors in the network register themselves and from which they can retrieve the other smart connectors currently available.

A single smart connector is an extended version of the Apache Jena Fuseki triple store that is not used for actually storing the triples, but for its reasoner to orchestrate the knowledge exchange process. This reasoner uses rules that are based on the configuration (i.e., capabilities) of all available smart connectors in the network about which every smart connector regularly retrieves updates from the service directory. Whenever a device publishes or requests data, the smart connectors make sure that it is received by or from the correct device. Mutually, the smart connectors use a combination of SPARQL and a publish/subscribe mechanism to communicate.

For example, a thermometer device would have a supply capability description, like the one shown in Figure 3(b). The thermostat requesting this information uses a demand capability description that looks identical, because both supply and demand capability descriptions follow the same SPARQL-like structure, multiple triples with variables for the data that it either demands or supplies.

The Knowledge Engine can use any ontology to structure the knowledge that it shares (both capability descriptions and Linked Data). For this experiment, SAREF was used, but it could also work with different ontologies like SOSA/SSN (Janowicz et al., 2019). By using the Knowledge Engine, we can validate SAREF by implementing it for the knowledge that is exchanged within its communications.

Figure 3 Two capability descriptions that can be used with the Knowledge Engine

```
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#>
@prefix sosa: <http://www.w3.org/ns/sosa/>
@prefix saref: <https://saref.etsi.org/core/>
@prefix ex: <http://example.org/>
```

```
?id rdf:type saref:FeatureOfInterest .
?id ex:hasName ?room .
?act saref:hasFeatureOfInterest ?id .
?act saref:actsUpon saref:OnOffState .
?act rdf:type ?command .
```

(a)

```
?id rdf:type saref:FeatureOfInterest .
?id ex:hasName ?room .
?obs rdf:type saref:Measurement .
?obs saref:isMeasurementOf ?id .
?obs saref:isMeasuredIn saref:TemperatureUnit .
?obs saref:hasValue ?temp .
```

(b)

4.3 Core scenario experiment

The smart connectors run on Raspberry Pi 4B devices, allowing easy configuration relevant to the device it is connected to with a config file.[4] The configuration requires a name and short description of the device and allows for inclusion of their capability descriptions. For this experiment, the smart device and smart connector are on one device with the devices being controlled with a Python script that implements the communication layer with the Apache Fuseki instance. A separate laptop runs the service directory and lastly, a Wi-Fi router allows the smart connectors to connect to each other and the service directory, as represented by the dashed lines in Figure 2.

The first Raspberry Pi functions as a smart thermometer. It disseminates information containing the temperature measurements made by the connected thermometer and the room it is located in. The graph pattern that reflects this information in SAREF can be found in Figure 3(b), `?room` is the name of the room and ?temp is the latest measurement of the temperature.

The second Raspberry Pi functions as a thermostat, it has an internal desired state for the temperature of this room which can be adjusted with the connected buttons. It demands the temperature of this room (knowledge demand in Figure 3(b) and supplies a command based on that current temperature (knowledge supply in Figure 3(a). If the current temperature is below the desired state the smart connector will send `saref:OnCommand for ?command`, if it is higher it will send `saref:OffCommand`.

The third Raspberry Pi represents a heating device, receiving heating commands for this room (knowledge demand in Figure 3(a). When the smart connector receives a `saref:OnCommand` it will turn on the heater (in this case the led light will turn on) and when it receives a `saref:OffCommand` it will turn it off.

When all three of these smart connectors communicate correctly they will be able to control the temperature in a room. Smart connector b will receive the temperature of the room from smart connector a, compare it with the desired temperature (which can be adjusted with its buttons), and based on that sends either an on or off command towards smart connector c.

4.4 Complete scenario experiment

In the core scenario the devices only use one term, either the temperature or the on/off state, from the messages that are communicated in the graph patterns. Thus we only need a small graph pattern to transmit the information (see Figure 3). When we use the complete SAREF *mapping* it also includes a lot of background information that can be used, allowing for more complicated scenarios. This can will be demonstrated by using the complete scenario and answering the fourth competency question.

The complete scenario includes an additional smart connector and device (in the shape of another Raspberry Pi), as can be seen in Figure 4. This new device represents seven different appliances of which it can supply the current energy consumption data through its smart connector. The energy consumption data was recreated by periodically sending one data point from each of the appliances from one residence in the OPSD data set (see Sub-sections 3.3 and 3.4) using the capability description in Figure 5 to supply the data.

Figure 4 Visualisation of the complete setup, the core experiment uses only the setup in the grey box and the complete experiment comprises all elements shown in this figure

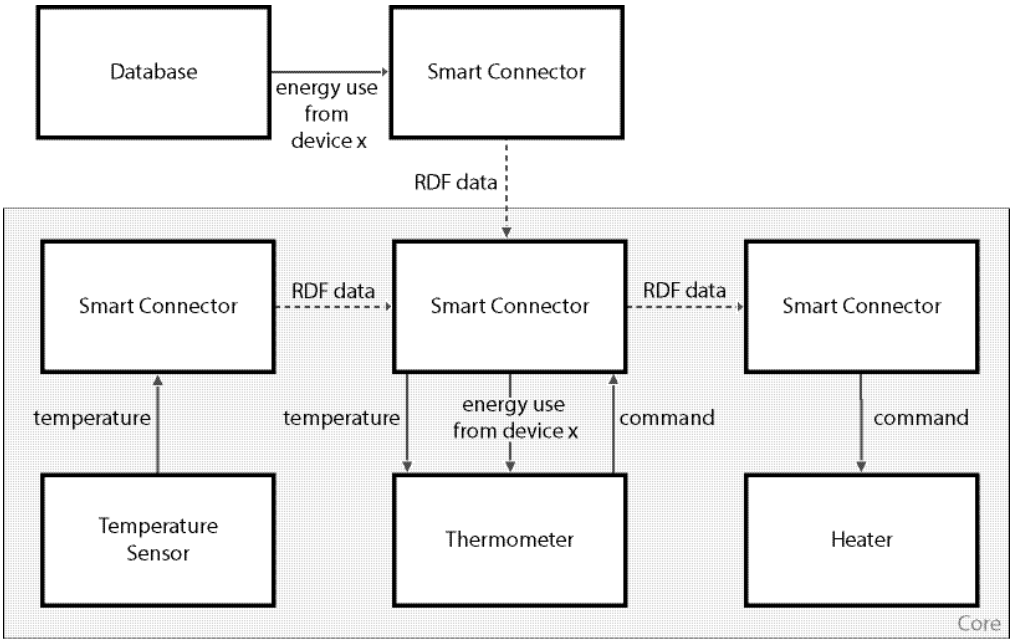


Figure 5 The new capability description used to communicate the measurements of power usage from a specific device

```
?measurement rdf:type saref:Measurement .
?measurement saref:hasValue ?value .
?measurement saref:isMeasuredIn om:kilowatt_hour .
?measurement saref:hasTimeStamp ?timestamp .
?measurement saref:isMeasurementOf> ?feature_of_interest .
?measurement saref:relatesToProperty> ?property .
?measurement saref:measurementMadeBy> ?device .
om:kilowatt_hour rdf:type saref:PowerUnit .
?property rdf:type saref:Power .
?property saref:relatesToMeasurement ?measurement .
?property saref:isPropertyOf ?feature_of_interest .
?property saref:isMeasuredByDevice ?device .
?feature_of_interest rdf:type saref:FeatureOfInterest .
?feature_of_interest saref:hasMeasurement ?measurement .
?feature_of_interest saref:hasProperty ?property .
?feature_of_interest ex:hasName ?room .
?device rdf:type ?device_type .
?device saref:makesMeasurement ?measurement .
?device saref:measuresProperty ?property .
?device saref:accomplishes ?task .
?device saref:hasFunction ?function .
?task rdf:type ?task_type .
?task saref:isAccomplishedBy ?device .
?function rdf:type saref:MeteringFunction .
```

Working under the assumption that devices with task type `Saref:Comfort` are devices that are only in use when a room is occupied we use the capability description in Figure 5 for the thermostat smart connector, a demand for energy consumption from all devices with the task type `Saref:Comfort`. With this new capability description the thermostat only receives energy consumption information from devices that are relevant to determining whether the room is occupied. It stores the two most recent energy consumption values of each device (which can be distinguished by the device URI) of the relevant task, and when a new temperature reading is received that would require an on command to be send to the heater it first confirms whether any of the devices have used energy, which it did if the two most recent energy consumption values for a device are different. Then, it will continue with sending the on command, otherwise the heater will remain off because the room is not occupied.

4.5 Answering the competency questions

In order to make the knowledge exchanges between the smart connectors required for the core scenario, two different capability descriptions are used, shown in Figure 3.

Using the information from these devices we are able to answer the first three competency questions. The graph patterns that are sent by the smart connector connected to the thermometer (see Figure 6(b)) answer the first question, the temperature of the room. The second competency question is answered internally in the smart connector connected to the thermostat, and the third competency question is solved by the thermostat smart connector sending its graph pattern that is received by the third smart connector that interprets and either turns the heater off or, in the case of Figure 6(a), on.

Figure 6 Example of a graph pattern from the thermostat smart connector (a) and the thermometer smart connector (b) answering the competency questions

```
ex:r1 rdf:type saref:FeatureOfInterest .
ex:r1 ex:hasName "Room1" .
ex:act1 saref:hasFeatureOfInterest ex:r1 .
ex:act1 saref:actsUpon saref:OnOffState .
ex:act1 rdf:type saref:OnCommand .
```

(a)

```
ex:r1 rdf:type saref:FeatureOfInterest .
ex:r1 ex:hasName "Room1" .
ex:obs1 rdf:type saref:Measurement .
ex:obs1 saref:isMeasurementOf ex:r1 .
ex:obs1 saref:isMeasuredIn saref:TemperatureUnit .
ex:obs1 saref:hasValue "17.0" .
```

(b)

By expanding to the complete scenario the system is able to answer the fourth competency question in two steps. First the graph pattern shown in Figure 5 includes the task of the device of which the measurements are sent. Within the thermostat smart connector the distinction is made between devices that perform task `Saref:Comfort` and devices that do not, thereby only considering devices that we consider to be indicative of the home being occupied. The thermostat takes the most recent values of only these devices in consideration when it is able to send the next command, and only does so when the house is indeed occupied, thereby answering the competency question correctly.

Using the complete scenario we are able to answer all four of the competency questions, by combining information coming from different devices, demonstrating that the information represented using SAREF is expressive enough to accomplish our scenario by making the devices in our setup interoperable.

5 Conclusions

The goal of this paper was to validate the practical use of SAREF with smart home data. This was done using two approaches.

The first approach showed how different data sets of smart home data can be mapped to Linked Data using SAREF. During the mapping of the synthetic data set we encountered two exceptions where the mapping was not able to use a pre-existing subclass, as shown in Sub-section 3.2, they were simple to add given the flexibility by the design of SAREF. On the other hand, this could potentially lead to incompatibility issues if different people create their own extensions for similar situations. To that end, ETSI has recently created a open community portal (saref.etsi.org) where SAREF users can submit their contributions that will then be considered in the official standardisation process to become part of new SAREF releases. The mapping of the OPSD data set was performed by creating rules using the YARRRML mapping language, to demonstrate that creating a mapping can also be simple for users that might not be very familiar with Linked Data.

The second approach was the implementation of a simple smart home scenario in which multiple devices need to share information in order to determine whether the heating device should be turned on or off. The knowledge engine, a promising framework that allows multiple IoT devices to exchange information in an interoperable way within a network, was used to exchange messages between the devices, while all the information in those messages represented with SAREF. Even though the data was not originally from the same data sets it was made interoperable due to how it was represented with SAREF.

Both the approaches show the viability of SAREF as an ontology to make smart devices interoperable, by showing that we can map realistic data without losing information, representing data with SAREF can be easily done (especially when using a mapping language) and the information can be used in a meaningful way to accomplish scenarios.

For the implementation of the scenario, all the devices used were Raspberry Pi's, allowing all the freedom to make changes to how they operate, replicating the scenario with actual retail smart devices could introduce new unforeseen problems that are worth investigating. More future work could recreate the scenario we presented with other interoperability frameworks or using different ontologies.

Acknowledgement

This work is part of the InterConnect project (interconnectproject.eu/) which has received funding from the European Union's Horizon 2020 research and innovation program under grant agreement No. 857237.

References

Bajaj, G., Agarwal, R., Singh, P., Georgantas, N. and Issarny, V. (2017) 'A study of existing ontologies in the IoT-domain', *arXiv preprint arXiv:1707.00112*.

Compton, M., Barnaghi, P., Bermudez, L., GarcA-Castro, R., Corcho, O., Cox, S., Graybeal, J., Hauswirth, M., Henson, C. and Herzog, A. et al. (2012) 'The SSN ontology of the W3C semantic sensor network incubator group', *Journal of Web Semantics*, Vol. 17, pp.25–32.

Daniele, L., Den Hartog, F. and Roes, J. (2015) 'Created in close interaction with the industry: the smart appliances reference (SAREF) ontology', *International Workshop Formal Ontologies Meet Industries*, Springer, pp.100–112.

Daniele, L., Strabbing, W., Roelofsen, B., Aalberts, A. and Stapersma, P. (2018) *Study on Ensuring Interoperability for Demand Side Flexibility*, Technical Report, European Commission. Doi: 10.2759/26799.

Dimou, A., Vander Sande, M., Colpaert, P., Verborgh, R., Mannens, E. and Van de Walle, R. (2014) 'RML: a generic language for integrated RDF mappings of heterogeneous data', *Proceedings of the 7th Workshop on Linked Data on the Web*.

ETSI TS (2016a) ETSI TS *103 375: SmartM2M; IoT Standards Landscape and Future Evolutions*.

ETSI TS (2016b) *ETSI TS 103 376: SmartM2M; IoT LSP Use Cases and Standards Gaps*.

ETSI TS (2020) *ETSI TS 103 264 V3.1.1, SmartM2M; Smart Applications; Reference Ontology and oneM2M Mapping*.

Gruninger, M. and Fox, M. (1995) 'Methodology for the design and evaluation of ontologies', *Proceedings of the Workshop on Basic Ontological Issues in Knowledge Sharing*, pp.1–11.

Heyvaert, P., De Meester, B., Dimou, A. and Verborgh, R. (2018) 'Declarative rules for linked data generation at your fingertips!', *European Semantic Web Conference*, Springer, pp.213–217.

Hsu, C.L. and Lin, J.C.C. (2016) 'An empirical examination of consumer adoption of internet of things services: network externalities and concern for information privacy perspectives', *Computers in Human Behavior*, Vol. 62, pp.516–527.

Janowicz, K., Haller, A., Cox, S.J., Le Phuoc, D. and Lefrançois, M. (2019) 'SOSA: a lightweight ontology for sensors, observations, samples, and actuators', *Journal of Web Semantics*, Vol. 56, pp.1–10.

Lassila, O., Swick, R.R., Wide, W. and Consortium, W. (1998) *Resource Description Framework (RDF) Model and Syntax Specification*, W3C Working Draft 20 July 1998.

Moreira, J., Daniele, L., Pires, L.F., Van Sinderen, M., Wasielewska, K., Szmeja, P., Pawlowski, W., Ganzha, M. and Paprzycki, M. (2017) 'Towards IoT platforms' integration semantic translations between W3C SSN and ETSI SAREF', *Joint International Conference on Semantic Systems Workshops*, Amsterdam, Netherlands.

Open Power System Data (2020) *Open Power System Data: Data Package Household Data*, Version 2020-04-15. Available online at: https://data.open-power-system-data.org/household_data/ 2020-04-15/

Rijgersberg, H., Van Assem, M. and Top, J. (2013) 'Ontology of units of measure and related concepts', *Semantic Web*, Vol. 4, No. 1, pp.3–13.

Sagar, S., Lefrançois, M., Reba, I., Khemaja, M., Garlatti, S., Feki, J. and Médini, L. (2018) 'Modeling smart sensors on top of SOSA/SSN and WoT TD with the semantic smart sensor network (S3N) modular ontology', *Proceedings of the 17th Internal Semantic Web Conference*, Monterey, USA, pp.163–177.

Van der Weerdt, R., De Boer, V., Daniele, L. and Nouwt, B. (2021) 'Validating SAREF in a smart home environment', in Garoufallou, E. and Ovalle-Perandones, M.A. (Eds): *Metadata and Semantic Research*, Springer International Publishing, Cham, pp.35–46.

W3C (2020) *W3C: Web of Things (WoT) Thing Description*. Available online at: https://www.w3.org/TR/2020/REC-wot-thing-description-20200409/

Websites

1 Image taken from: https://saref.etsi.org/core/v3.1.1/#Figure_1

2 https://github.com/RoderickvanderWeerdt/IoT_data_SAREF_mappings

3 https://github.com/RoderickvanderWeerdt/SAREF-YARRRML-mappings

4 https://jena.apache.org/documentation/fuseki2/fuseki-configuration.html