Towards IoT platforms' integration: Semantic Translations between *W3C SSN and ETSI SAREF*

João Moreira* Laura Daniele† Luis Ferreira Pires*

Marten van Sinderen* Katarzyna Wasielewska‡ Pawel Szmeja‡

Wiesław Pawłowski§ Maria Ganzha‡ Marcin Paprzycki‡

ABSTRACT

Several IoT ontologies have been developed lately to improve the semantic interoperability of IoT solutions. The most popular of these ontologies, the W3C Semantic Sensor Network (SSN), is considered an ontological foundation for diverse IoT initiatives, particularly OpenIoT. With characteristics similar to SSN, the ETSI Smart Appliances REFerence (SAREF) ontology evolved from the needs of smart home solutions to common requirements of IoT. Some IoT solutions rely on platform-specific ontologies and their integration requires mechanisms to align these ontologies. In this paper we discuss the ontology alignment between SSN and SAREF, identifying mapping alternatives and proposing basic mappings that can be re-used to define more complex ones. We introduce here an initial specification of the semantic translations from the main elements of SSN to SAREF, which includes classes, object properties and data properties. The alignment will be used in a semantic matching process leveraging the semantic mediator component of the INTER-IoT project. An initial evaluation of the translation was executed by translating the wind sensor (Vaisala WM30), an example provided by the W3C, from SSN to SAREF. This initial evaluation demonstrates the coherence and feasibility of the proposed mappings.

*University of Twente, Enschede, The Netherlands.
{j.luizrebelomoreira@utwente.nl, l.ferreirapires@utwente.nl, m.j.vansinderen}@utwente.nl

†Netherlands Organisation for Applied Scientific Research, TNO.
laura.daniele@tno.nl

‡Systems Research Institute, Polish Academy of Sciences, Warsaw, Poland
{katarzyna.wasielewska, pawel.szmeja, maria.ganzha, marcin.paprzycki}@ibspan.waw.pl

§Faculty of Mathematics, Physics, and Informatics, University of Gdańsk, Poland
w.pawlowski@inf.ug.edu.pl

SEMANTICS '2017 Amsterdam, The Netherlands

DOI: DOI

CCS Concepts

•**Information systems** → **Semantic web description languages;**

Keywords

Semantic interoperability, internet-of-things, ontology alignment, ETSI SAREF, W3C SSN

1. INTRODUCTION

Over the past few years, numerous IoT ontologies were proposed to improve the semantic interoperability of IoT artifacts, i.e. the common understanding capabilities of platforms, devices, gateways, applications and networks involved in IoT solutions [8]. In this context, the W3C Semantic Sensor Network (SSN) is considered an ontological foundation for the IoT, covering the application of diverse types of sensors, widely used by initiatives as OpenIoT [12] and INTER-IoT [7]. Recently, the Smart Appliances REFerence (SAREF) ontology has evolved from the smart appliances domain (e.g. smart ovens and refrigerators) [4] to cover other characteristics of the IoT domain [6], being created in close interaction with the smart home market [5]. It is grounded on 47 "semantic assets", i.e. standards, proprietary data models, protocols and other ontologies, as SSN.

IoT platforms enable software engineers to bridge the gap between device sensors and connected applications through suites of components, which can include semantic technologies, e.g. FIWARE with the Sense2Web linked-data generic enabler (GE) and OpenIoT with its own ontology extended from SSN. The Inter-IoT project[1] aims at designing, implementing and experimenting with voluntary interoperability among heterogeneous IoT platforms. The project is driven by use cases from (e/m)Health and transportation/logistics at the port of Valencia, which require semantic integration among IoT artifacts relying on SSN and SAREF.

A challenge to enable semantic integration between IoT artifacts relying on SSN and SAREF, is how to translate from one ontology to another, i.e. align these ontologies. To deal with this problem, the development of an ontology alignment between SSN and SAREF is required, i.e. finding and mapping the correspondences between entities (atomic) or groups of entities and sub-structures (complex). An approach to implement ontology alignment is semantics trans-

[1]www.inter-iot-project.eu

lation, which is a process that transforms some information described semantically, in terms of a source ontology, to information described in terms of a target ontology [9].

In this paper we introduce mappings among the main elements for the semantic translations from SSN 1.0 to SAREF 2.0. We adopted a model-driven engineering methodology to develop these semantic translations which considers specification and implementation. The specification will be used for configuration and deployment in the Inter-Platform Semantic Mediator (IPSM) of INTER-IoT. We evaluated these mappings by demonstrating how the SSN representation of the WM30 wind sensor, which is an example provided by the W3C, is translated to SAREF. The semantics of the generated ontology are verified after the execution of the translations to determine whether they were maintained or not.

Section 2 describes the background research with an overview of the IPSM, the SSN and SAREF ontologies. Section 3 describes the methodology used to develop the semantic translations. Section 4 introduces the mappings from SSN to SAREF, describes the evaluation of the mappings and discusses the challenges. Section 5 concludes this work by presenting the lessons learned and future work.

2. BACKGROUND

2.1 INTER-IoT semantic mediator

The main goal of the H2020 project Interoperability of Heterogeneous IoT Platforms (INTER-IoT) [7, 10] is to design, implement and validate a framework that will allow interoperability between heterogeneous IoT platforms across the transportation (logistics) and (e/m)Health domains. The use case scenarios are based on real world situations that need integration among IoT platforms, as the port authority IoT platform and the haulier IoT platform, with systems as the container terminal system and the port management. The (e/m)Health domain scenarios aims at improving interoperability among IoT artifacts for patient monitoring, e.g. body sensor networks, wearable and non-wearable devices. Some of these IoT artifacts rely on SSN, such as the OpenIoT platform. In the scenario of detecting emergencies at the port by monitoring drivers' vital signs with medical wearable devices, semantic integration is required between IoT artifacts based on SSN and smart appliances based on SAREF, e.g. building sensors for vehicle collision detection, security and electrical systems. INTER-IoT is currently developing the Inter-Platform Semantic Mediator (IPSM) tool. IPSM is a software tool that follows the semantic interoperability design patterns identified in INTER-IoT [7], and is intended to be used as part of the translation process defined in the methodology (INTER-Meth). The process of achieving semantic interoperability involves the following steps: (i) lift semantics to a common format and make it explicit[7]; (ii) develop, or choose, a central modular ontology; (iii) prepare uni-directional alignments between ontologies of communicating artifacts and the central ontology; (iv) establish a multi-channel (1-1, 1-many, many-1) communication architecture to facilitate translations in all needed contexts, with the central ontology as intermediary.

INTER-IoT provides its own alignment format, based on an alignment API with a declarative ontology alignment language (XML-based), to represent interconnections between semantic data of multiple ontologies. IPSM utilizes alignment files and provides a multi-channel environment for any artifact. Pairs of uni-directional alignments between the central ontology and artifact ontology are used to translate messages to and from the central ontology. This enables connection of new artifacts without jeopardizing the existing channels, and requires each participant to provide only a pair of alignments. While complete ontologies are used to build semantic understanding, only conversation-specific alignments are stored and used for actual translations. Ontology alignments and translation channels can be managed through the REST Manager. Because of space limitations in this paper we omit other related work, which can be found in our prior publications that cover the state-of-the-art in this area [7, 8, 9, 10].

2.2 W3C Semantic Sensor Network

The W3C Semantic Sensor Network (SSN)[2] [2] is an ontology developed by W3C, (current published version 1.0, 2011). It provides a comprehensive framework to describe sensors, devices, observations, measurements and other terms, enabling reasoning of individual sensors and the connection of sensors, such as wireless networks. SSN 1.0 is grounded in a set of existing ontologies and standards, such as CSIRO, SWAMO, SEEK Extensible Observation, SemSOS and OGC SensorML [8]. The main concept of SSN is the *Sensing Device*, which is a sensor that reports measurements and observations of real world phenomena. A sensor is different in nature from other types of devices, e.g. actuators, because of its event-based behavior, which requires temporal relationships. SSN enables reasoning, which can facilitate the development of advanced applications, for example, by reasoning about sensor measurements, considering constraints as power restriction and limited memory. It consists of 10 modules, representing 41 concepts and 39 object properties. It inherits 11 concepts and 14 object properties from the foundational ontology DOLCE-UltraLite (DUL)[3]. In this paper we cover the following modules:

DUL module[4]: represents the foundational categorization of *Designed Artifact*, *Method*, *Physical Object*, *Quality*, *Region* and *Situation*. For example, a *Sensing Device* (Measuring module) is a *Designed Artifact* and a *Physical Object*, which observes a *Property* (Skeleton module). A *Property* is an observable *Quality* of an *Event* or *Object*, i.e. "an aspect of an entity that is intrinsic to and cannot exist without the entity and is observable by a sensor".

Skeleton module: represents the most basic concepts regarding sensors, as *Sensor*, *Sensing*, *Property* and *Observation*. A *Sensor* may be a physical device implementing *Sensing*, i.e. it has a sensing method observing some *Property*. "*Sensing* is a process that results in the estimation, or calculation, of the value of a phenomenon".

Measuring module: covers the elements *Sensing Device* and *Sensor DataSheet*. The prior is the main element of SSN. The former represents the data sheet specifications of a sensor. Usually, the properties of a sensor are recorded directly with *hasMeasurementCapability* property of a *Sensor*.

System module: represents the *System* concept as a *Physical Object* (DUL) composed by sub-systems (*hasSubSystem*), which has deployment(s) (*hasDeployment*), operat-

[2]https://www.w3.org/2005/Incubator/ssn/ssnx/ssn

[3]http://ontologydesignpatterns.org/wiki/Ontology:DOLCE+DnS_Ultralite

[4]https://www.w3.org/2005/Incubator/ssn/wiki/DUL_ssn

ing range(s) (*hasOperatingRange*) and location(s) relative to other entities (*onPlatform*).

Measuring Capability module: represents core concepts of SSN, i.e. properties and capabilities of measurements, such as *Accuracy*, *MeasurementProperty* and *MeasurementCapability*. Relevant object properties are the *hasMeasurementCapability* and *hasMeasurementProperty*. *MeasurementCapability* represents a characteristic of a sensor's observations or ability to make observations (e.g. accuracy and range). *MeasurementProperty* represents the collection of measurement properties and environmental conditions in which those properties hold.

Device module: covers *Device*, which is a physical piece of technology (a "system in a box") and can be composed of other (smaller) devices and software components.

The W3C SSN Incubator Group created an example of the use of SSN by describing the *Vaisala WM30 wind sensor*[5]. It describes the measurement capabilities, power supply and operating and survival properties based on the technical specification of the measurement of wind direction and speed. This example was initially reported in [3], which gives a precise description of the WM30 sensor. In this paper we updated these axioms, since WM30 ontology main definitions and restrictions are different, a consequence of the changes in the TBox of SSN until the current published version. These axioms represent the Vaisala WM30 as a *Sensing Device* (therefore a *Device* (*System*) and a *Sensor*), composed by (*hasSubSystem*) WM30 particular sensors for wind direction and wind speed, being able to measure (*observes*) wind direction (*WM30_WindDirection)* and speed (*WM30_WindSpeed*):

$$\texttt{WM30:Vaisala\_WM30} \sqsubseteq \texttt{ssn:SensingDevice}$$

$$\texttt{WM30:Vaisala\_WM30} \sqsubseteq \exists\ \texttt{ssn:hasSubSystem}\,.\,\texttt{WM30:WM30\_WindDirection}$$

$$\texttt{WM30:Vaisala\_WM30} \sqsubseteq \exists\ \texttt{ssn:hasSubSystem}\,.\,\texttt{WM30:WM30\_WindSpeed}$$

$$\texttt{WM30:Vaisala\_WM30} \sqsubseteq \exists\ \texttt{ssn:observes}\,.\,\texttt{WM30:\_WindDirection}$$

$$\texttt{WM30:Vaisala\_WM30} \sqsubseteq \exists\ \texttt{ssn:observes}\,.\,\texttt{WM30:\_WindSpeed}$$

Currently the W3C is updating the entire SSN ontology towards a new version (2.0), which is available as a public draft document prepared by the W3C and the Open Geospatial Consortium (OGC)[6]. In this new version, a new ontology is introduced, namely the Sensor Observation Sampling Actuator (SOSA), absorbing a number of classes and properties from SSN 1.0, such as *Sensor*, *Observation* and *observes*. SOSA is aligned with DUL and SSN 2.0 is aligned with SOSA. SOSA is also aligned with the foundational ontologies BFO and PROV-O[7]. Most important, the compatibility of our work with SSN 2.0 is not compromised because the specification provides alignments of SSN 2.0 with SSN 1.0, including complex ones.

2.3 ETSI Smart Appliances REFerence

Recently, the European Telecommunications Standards Institute (ETSI) along with the European Comission (EC), the Netherlands Organisation for Applied Scientific Research (TNO), the Universidad Politécnica de Madrid (UPM) and other partners, developed the Smart Appliances REFerence (SAREF) ontology [4, 5][8]. At first this ontology was built as a reference model targeting smart appliance solutions for the smart home domain[9]. However, SAREF has evolved to cover the IoT domain in general, being acknowledged by the EC as the "first ontology standard in the IoT ecosystem, and sets a template and a base for the development of similar standards for the other verticals to unlock the full potential of IoT" [6]. The SAREF ontology provides building blocks that enable re-utilization of different parts of the ontology according to specific requirements. The new version SAREF 2.0[10] brings a number of changes towards this evolution, including new alignments with OneM2M for services' provision of smart things.

SAREF facilitates the matching of existing assets, since it was developed based on standards, ontologies, data models and protocols of the IoT domain, providing a high-level mapping of them (available in SAREF's first interim study report). One of these assets is SSN, which inspired the definition of the main elements of SAREF, namely *Device*, *Sensor*, *Unit of Measure* and *Time/Duration*, according to the high-level mappings provided in the SAREF initial documentation [4]. A *Device* (e.g. a *Sensor*) represents tangible objects designed to accomplish one or more functions in diverse types of locations (e.g. households and buildings). For example, a *Sensor* has *Function* of type *Sensing function*. The SAREF ontology offers a list of basic functions that can be combined towards more complex functions in a single device. For example, a *Switch* can offer an *Actuating function* of type "switching on/off" and a *Sensing function* of type *Light Sensor*, so if there is illumination in the environment then the switch turns off the light. Each *Function* has some associated *Commands*, which can also be picked up as building blocks from a list. For example, the "switching on/off" is associated with the commands "switch on", "switch off" and "toggle". Depending on the *Function(s)* it accomplishes, a device can be found in some corresponding *State(s)* that are also listed as building blocks.

The composition of a *Device* can be represented through the *saref:consistsOf* self-relationship, e.g. the WM30 wind sensor (a device) can be defined as a composition of wind direction and wind speed sensors. A *Device* measures a specific property, represented by the object property *saref:measuresProperty* to a *Property*. For example, a *SmokeSensor* (*Sensor*) measures *Smoke* (*Property*), analogously a *WindSensor* measures *Wind*. Regarding a measurement observed by a sensor in time, SAREF represents it through the *saref:makesMeasurement* object property of a *Device* to *Measurement(s)*, representing the relation between a device and the measurements it makes. A *Measurement* element links of the value of the *Measurement*, its *Unit of Measure* and the *Property* to which it relates.

Further, a *Device* offers a *Service*, which is a representation of a *Function* to a network that makes the function

[5]https://www.w3.org/2005/Incubator/ssn/ssnx/meteo
[6]https://www.w3.org/TR/vocab-ssn
[7]https://www.w3.org/2015/spatial/wiki/SOSA_Ontology
[8]http://ontology.tno.nl/saref/
[9]https://ec.europa.eu/digital-single-market/blog/new-standard-smart-appliances-smart-home
[10]https://ec.europa.eu/digital-single-market/en/news/new-version-machine-2-machine-standard-smart-appliances-introduced-etsi

discoverable, registerable and remotely controllable by other devices in the network. A *Service* can represent one or more functions. A *Service* must specify the *Device* that offers the *Service*, the function(s) to be represented, and the (input and output) parameters necessary to operate the service, which is supported by the recent ontology alignments with the OneM2M ontology.

3. METHODOLOGY

We surveyed tools for ontology alignment [9], describing the conceptual differences of alignment, matching, merging, mapping and semantic translations. Here we describe the semantic translations between SSN to SAREF and, thus, a methodology is required for implementing and evaluating these translations. Our methodology for semantic translations development follows a common software engineering approach, considering specification and implementation phases during the design time of the ontology alignment. The specification describes in natural language the possible mappings and the involved rules, linking the original ontology to the generated ontology. This methodology is based on a typical pattern Model-Driven Engineering (MDE) [1] to specify transformations in terms of a source and a target meta-model, as illustrated in Figure 1.

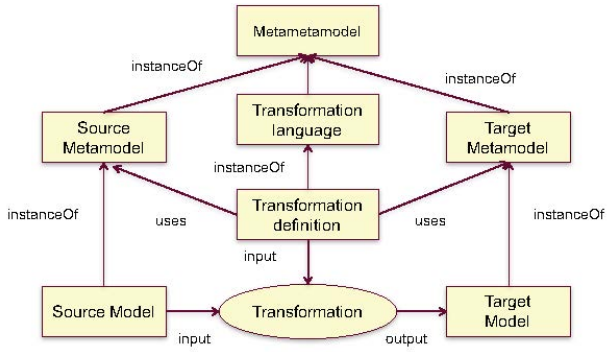

Figure 1: Model transformations in MDE

The set of mappings from SSN to SAREF is a transformation definition, which is an instance of a transformation language. These mappings are a definition of transformations of the instance level of these ontologies, e.g. the transformations are able to transform from the WM30 ontology (an instance of SSN) to an equivalent instance of SAREF. For the implementation of these mappings, a transformation language could be the library Apache JENA[11] (Java) and the mappings (transformation definition), which uses the SSN (source) and SAREF (target) meta-models, an instance of JENA implementation. The mappings and a message according to SSN (source) are used as input by the transformation runtime environment, e.g. Java Runtime Environment (JRE), which produces a SAREF (target) ontology with similar semantics as the original.

We use a specification strategy to avoid conceptual errors during the implementation, which is a common issue in software engineering. Moreover, directly implementing the mappings forces a preliminary choice of the technology of the transformation runtime, which couples the mappings with a technology, thus, limiting the reuse of the mappings. Our ontology-driven conceptual modeling strategy [11] can address this issue by leveraging the MDE approach with a specification artifact targeted to human readers, i.e. the transformation developers. For specification, the transformation runtime is not considered. For the transformation definition, an approach based on natural language enriched with pseudo-code is used instead of a formal language, similar to the OASIS EDXL-TEP and HL7 (syntactic interoperability standards) transformations[12]. This approach enables to (re)use the specification for different strategies to implement ontology alignment. The implementation of our mappings, i.e. the transformation definition, is planned to be represented as configuration files (XML format), which is an instance of the IPSM configuration schema (transformation language). However, this is an ongoing activity and this paper only covers the specification artifact, presented as pseudo-code.

A MDE transformation can be developed as an unidirectional or a bi-directional transformation. The best practices show that, when creating a bi-directional transformation with a high number of meta-model elements, a cyclical process should be used. The first step is to specify an unidirectional transformation with a selected sub-set of meta-model elements to be covered and validate with simulations. If errors are found then they should be fixed and the transformations should be validated again until the mappings correctly generate the intended model. The second step is to specify the opposite direction with the same elements of the first step, and validate/correct with simulations. The third step is to evaluate whether meaning is lost when executing the transformations in sequence, i.e. check how x is different to $\mathrm{T}(\mathrm{T}(x)_{\mathrm{A>B}})_{\mathrm{B>A}}$, where $\mathrm{T}(a)_{\mathrm{A>B}}$ produces the model b (meta-model B) from model a (meta-model A), and $\mathrm{T}(b)_{\mathrm{B>A}}$ the opposite direction. This step supports possible conceptual errors, enabling the early correction of the mappings before implementing. Once an acceptable level of the quality of the bi-directional transformation specification is achieved, the transformations can be implemented. This is an interactive process that must be executed while there are elements to be addressed, as in agile development methods. In this paper we present the first step applied from SSN to SAREF with three terms from SSN.

The rationale for specification are based on an ontological analysis of the SSN and SAREF TBox, i.e. an analysis of the statements that describe their conceptualization. Since SSN is extended from DUL, and DUL is the lightweight foundational ontology of DOLCE, we used DOLCE's categories as a reference to map from SSN to SAREF. For example, the object property *ssn:hasSubSystem* is a *DUL:hasPart*, which means "a schematic relation between any entities". Based on this definition, we sought for in SAREF the possible relations that have the same semantics for this mapping, finding that *saref:consistsOf* has the same meaning: "a relationship indicating a composite entity that consists of other entities (e.g., a temperature/humidity sensor that consists of a temperature sensor and a humidity sensor)". Although this process is quite time consuming, error-prone and automatic techniques based on natural language processing (NLP) are commonly used for finding ontology similarity [9], such on-

[11]https://jena.apache.org/

[12]http://docs.oasis-open.org/emergency/TEP-HL7v2-transforms/v1.0/TEP-HL7v2-transforms-v1.0.html

tological analysis can produce a more semantically assertive set of ontology alignments that does not rely only on the terms, but also on their descriptions, thus, their meaning. Therefore, we produced an initial documentation regarding the classes and object properties of both ontologies and their possible relations, analyzing each class/object property according to their definitions.

Here we present an initial evaluation of the specification created, simulating unidirectional semantic translation from WM30 ontology (SSN) to SAREF. In general, the result from the translations must represent the WM30 wind sensor with SAREF, keeping similar semantics of the original. Therefore, the goal of this evaluation is to check the semantic similarity between the original and the final ontologies. Here we used an approach based on competency questions to measure this similarity. A list of competency questions is presented, and each question is answered by navigating the elements of both ontologies. The answers are compared to verify whether the semantics are maintained after the simulation. Intentionally, the competency questions were conceived according to the expressiveness of the SSN WM30 ontology, targeting its main elements, as the different measurement capabilities described in the technical specifications[13]. For example, the accuracy of the WM30 wind speed sensor (within a range from 0.4 to 60 m/s) is +- 0.3 m/s for wind speed lower than 10 m/s and +-2% of variance for wind speed higher than 10 m/s. At first, we answered each question based on the original ontology (in SSN) and, then, we answered the same question for the generated ontology (SAREF). Second, we compare whether the semantics of the response is similar to each other, i.e. if the semantics are lost or maintained. The competency questions are:

CQ01. What are the characteristics of the WM30 sensor?
CQ02. What is the composition of this sensor, i.e. what are the parts of WM30?
CQ03. What measurement properties this sensor performs?
CQ04. What are the accuracy, delay distance, starting threshold and damping ratio of WMS30 wind direction sensors?
CQ05. What measurement range constraints differentiate these types of wind direction sensors?

4. SEMANTIC TRANSLATIONS

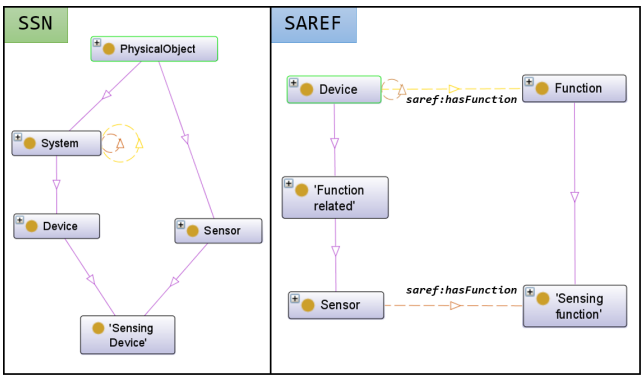

Figure 2: Main elements of SSN and SAREF

[13]http://www.vaisala.fi/Vaisala%20Documents/Brochures%20and%20Datasheets/WM30-Datasheet-B210384EN-B-LoRes.pdf

4.1 Mappings: SSN to SAREF

According to the methodology used, the mappings between SSN and SAREF were specified through an ontological analysis of their TBox, i.e. concepts and roles definitions (predicates) with logical operations. A study was made on how SSN and SAREF describe the characteristics of sensors, including their capabilities of observation. The mappings follow a logic sequence according to the main elements and similar structures of SSN and SAREF. Here we detail only the mappings from SSN to SAREF as a first step towards the creation of the bi-directional translations. For each mapping a code snippet is presented as a pseudo-code to illustrate the algorithm for the creation of the SAREF-based ontology. This pseudo-code include an *IN* representing the input SSN element(s). When creating a new SAREF class or property, *var* is used and *createTriple* function creates a triple (class, object property, class).

M01. ssn:SensingDevice -> saref:Sensor

While the main element of SSN is the *Sensing Device*, a subclass of *Device* and *Sensor*, in SAREF the main element is *Device*, which can be specialized as a *Sensor* related to a *Sensing Function*. The characteristics of *ssn:SensingDevice*, inherited from *ssn:Sensor* and *ssn:System*, are mapped to *saref:Sensor*, inheriting *saref:Device* properties, including the relationship with the *saref:SensingFunction* (*saref:hasFunction*). Figure 2 illustrates the elements involved in this mapping. Notice that, indirectly, this mapping also transforms from *ssn:Sensor* to *saref:Sensor* if the *ssn:Sensor* is a *ssn:SensingDevice*. All RDFS properties are copied from *ssn:SensingDevice* to *saref:Sensor*.

```
1 IN: ssn_sensingDevice
2 var saref_sensor = saref:Sensor
3 saref_sensor.rdfs = ssn_sensingDevice.rdfs
4 var saref_function = saref:SensingFunction
5 var saref_hasFunction = saref:hasFunction
6 createTriple(saref_sensor saref_hasFunction
      saref_function)
```

Listing 1: Pseudocode snippet for M01

M02. ssn:hasSubSystem -> saref:consistsOf

```
1  IN: ssn_sensingDevice, saref_sensor
2  for each ssn_system in
       ssn_sensingDevice.ssn:hasSubSystem
3    var saref_device_component = saref:Device
4
5    if ssn_system is ssn:SensingDevice then
6      saref_device_component = Map(M01,
           ssn_system)
7    else if ssn_system is ssn:Device then
8      saref_device_component.rdfs =
           ssn_system.rdfs
9
10   var saref_consistsOf = saref:consistsOf
11   createTriple(saref_sensor saref_consistsOf
         saref_device_component)
12 end for
```

Listing 2: Pseudocode snippet for M02

After executing M01, M02 checks the composition relationship of a device, i.e. the components that are part of a device. In SSN, the object property *ssn:hasSubSystem* relating two *ssn:System* represents this relationship. In SAREF,

the object property *saref:consistsOf* plays this role, relating two *saref:Device* in a similar way, both illustrated in Figure 2 as a self-relationship. Therefore, when *ssn:hasSubSystem* is used between the *ssn_sensingDevice* (from M01) and a *ssn:System*, which must be a *ssn:Device* or a *ssn:SensingDevice*, then it is mapped to *saref:consistsOf* object property of the *saref_sensor* (created in M01). If the device component is a *ssn:SensingDevice*, then a recursive algorithm is used by applying M01 to it. If the device component is a *ssn:Device*, then it is created a *saref:Device*.

M03. ssn:observes -> saref:measuresProperty
After executing M01 and M02, M03 maps the measurement property which the sensor is able to observe. For example, a wind sensor is able to observe both wind direction and wind speed. Therefore, the *ssn:observes* of a *ssn:SensingDevice* is mapped to *saref:measuresProperty* of a *saref:Device*. These object properties relate to *ssn:Property* and *saref:Property*, respectively. Therefore, this mapping also includes the creation, if it does not exist, of the *ssn:Property*. At last, this mapping needs to create the relationship back from the *saref:Property* to the *saref:Device* through the object property *saref:isMeasuredByDevice*. The code snippet below illustrates this mapping.

```
1  IN: ssn_sensingDevice, saref_sensor
2  for each p in ssn_sensingDevice.observes
3    var ssn_property = p
4    var saref_property = saref:Property
5    saref_property = GetPropertyInSAREF(
         ssn_property)
6
7    if not isnull(saref_property) then
8      saref_property.rdfs = ssn_property.rdfs
9
10   var saref_measuresProperty =
         saref:measuresProperty
11   var saref_sensor saref_measuresProperty
         saref_property
12   var saref_isMeasuredByDevice =
         saref:isMeasuredByDevice
13   createTriple(saref_property
         saref_isMeasuredByDevice saref_sensor)
14 end for
```

Listing 3: Pseudocode snippet for M03

4.2 Evaluation

As described in the methodology section, an execution of the mappings presented above was simulated having the WM30 wind sensor example (according to SSN) as input. The output of this translation, i.e. the WM30 ontology according to SAREF, was produced and made available [14]. The answers of the competency questions (see section 3) were produced by navigating the generated ontology with support of an ontology editor (Protegé and TopBraid Composer). The competency questions were responded as:
CQ01. In the original ontology (SSN), the main characteristics of the WM30 sensor can be extracted by starting the navigation in the *WM30:Vaisala_WM30* element, which is a *ssn:SensingDevice*, inheriting the properties from *ssn:Device* and *ssn:Sensor*. Besides, the *rdfs:comment* with a general description of this wind sensor, it represents that the sensor is composed by two sensors (*WM30:WM30_WindDirection* and *WM30:WM30_WindSpeed)*, one for wind direction and another for wind speed (both *ssn:Sensor*). Moreover, WM30 sensor can measure (observe) the types (properties) *WM30:WindDirection* and *WM30:WindSpeed* (both *ssn:Property*). Figure 3 (top) illustrates these properties.

In the generated ontology (SAREF), according to M01, *WM30:Vaisala_WM30* element is created as a *saref:Sensor*. A *saref:SensingFunction* is created and the *WM30:Vaisala_WM30* element linked to it through *saref:hasFunction* property. According to M02, the composition of the sensor *WM30:Vaisala_WM30* is derived from the *ssn:hasSubSystem* properties, i.e. *WM30:WM30_WindDirection* and *WM30:WM30_WindSpeed*. M03 produced the measurement properties of the sensor from the *ssn:observes element*, i.e. *saref:measuresProperty* to *WM30:WindDirection* and *WM30:WindSpeed*. Figure 3 illustrates the result *WM30:Vaisala_WM30)* as a *saref:Sensor*. Therefore, by navigating to *WM30:Vaisala_WM30*, a *saref:Sensor*, it is possible to respond this competency question in a similar way from the original, i.e. the semantics is completely maintained.

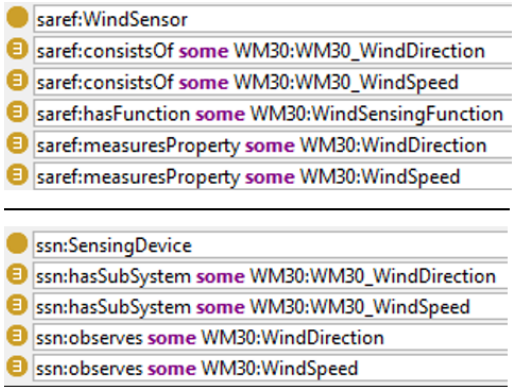

Figure 3: *ssn:SensingDevice* and *saref:Device*

CQ02. In the original ontology (SSN), the composition of WM30 sensor can be extracted by navigating from the *WM30:Vaisala_WM30* element to the *ssn:hasSubSystem* properties, i.e. *WM30:WM30_WindDirection* and *WM30:WM30_WindSpeed*, both *ssn:Sensor*. The WM30 wind direction sensor (*WM30:WM30_WindDirection*) can have one wiper (*WM30:WMS301*) or two (*WM30:WMS302*). The same structure is generated in SAREF, resulted from M02, having *WM30:WM30_WindDirection* and *WM30:WM30_WindSpeed* created as *saref:Device*, linked through the object property *saref:consistsOf*. It is possible to achieve the same structure of *WM30:WM30_WindDirection* and *WM30:WM30_WindSpeed*, including the specialization to one or two wipers, by navigating from *WM30:Vaisala_WM30* (*saref:Sensor*) through *saref:consistsOf*. This competency question is responded in a similar way from the original (semantics maintained).
CQ03. In the original ontology (SSN), the measurement properties of Vaisala WM30 sensor can be extracted by navigating from the *WM30:Vaisala_WM30* element through the *ssn:observes* properties, i.e. *WM30:WindDirection* and *WM30:WindSpeed*, both *ssn:Property*. WM30 original example also uses an ontology of Quantity Kinds (http://purl.oclc.org/NET/ssnx/qu) through the element *qu:QuantityKind* as *ssn:Property*. This element provides a taxonomy of quality dimensions, making use of *dim:Angle* for *WM30:WindDirection*

[14]https://github.com/jonimoreira/SSN-SAREF)

and *dim:VelocityOrSpeed* for *WM30:WindSpeed*. The same structure is generated in SAREF, resulted from M03, as *saref:Property*. Therefore, similar to CQ01 and CQ02, the semantics is completely maintained.

CQ04. The specification of the WMS30 wind direction sensor describes the accuracy, delay distance, starting threshold and damping ratio of this sensor. For example, accuracy can vary from -3 to 3 degrees, while the delay distance is 0.6 meters and the starting threshold is lower than 1.0 m/s. In the original ontology (SSN), the measurement capabilities of each wind direction and speed components of Vaisala WM30 sensor can be extracted by navigating from the *WM30:Vaisala_WM30* element through the *ssn:hasSubSystem* properties, i.e. *WM30:WM30_WindDirection* and *WM30:WM30_WindSpeed*, both *ssn:Sensor*, as described in CQ02. The WM30 wind direction sensor with one wiper (*WM30:WMS301*) has measurement capability a *WM30:WM30_WindDirection_MeasurementCapability_WMS301* (similar for *WM30:WMS302*). A *WM30:WM30_WindDirection_MeasurementCapability_WMS301* is a *WM30:WM30_WindDirection_MeasurementCapability*, which describes the ranges supported (restrictions) for accuracy, delay distance, starting threshold and damping ratio, illustrated in the axioms of Figure 4. *ssn:MeasurementCapability* is a *ssn:Property*. Regarding the accuracy, SSN provides naively the *ssn:Accuracy* element which represents the accuracy of all involved sensors, extracted with a simple SPARQL query (*SELECT * WHERE { ?a ?b ssn:Accuracy. }*) .

```
ssn:forProperty only WM30:WindDirection
ssn:hasMeasurementProperty some (WM30:DampingRatio
 and (ssn:hasValue some (DUL:Amount
 and (DUL:hasRegionDataValue value 0.3))))
ssn:hasMeasurementProperty some (WM30:DelayDistance
 and (ssn:hasValue some (DUL:Amount
 and (WM30:unitOfMeasure some {unit:metre})
 and (DUL:hasRegionDataValue value 0.6))))
ssn:hasMeasurementProperty some (WM30:OverShootRatio
 and (ssn:hasValue some (DUL:Amount
 and (DUL:hasRegionDataValue value 0.4))))
ssn:hasMeasurementProperty some (WM30:StartingThreshold
 and (ssn:hasValue some (WM30:ValueRange
 and (WM30:unitOfMeasure some {unit:metrePerSecond})
 and (WM30:hasRangeMaxValueExclusive value 1.0))))
ssn:hasMeasurementProperty some (ssn:Accuracy
 and (ssn:hasValue some (WM30:ValueRange
 and (WM30:unitOfMeasure some {unit:degreeUnitOfAngle})
 and (WM30:hasRangeMaxValueExclusive value "3.0"^^xsd:float)
 and (WM30:hasRangeMinValueExclusive value "-3.0"^^xsd:float))))
ssn:ofFeature only WM30:WindFeature
```

Figure 4: Measurement capabilities of WM30

In the generated ontology (SAREF), these measurement capabilities were lost because there is not a similar structure of *ssn:MeasurementCapability* in SAREF, thus, no mappings were added to consider the *ssn:hasMeasurementCapability* object property of *ssn:Sensor*. This question could not be responded with the generated ontology (semantics was lost).

CQ05. In the original ontology (SSN), the measurement range constraints differentiating 301 and 302 wind direction sensors can be extracted by analyzing the restrictions of *WM30:WM30_WindDirection_MeasurementCapability_WMS301* and *WM30:WM30_WindDirection_MeasurementCapability_WMS302* regarding the object property *ssn:hasMeasurementProperty*. The first restricts the measurement range from 0 to 355 degrees, while the second restricts from 0 to 360 degrees. This question could not be responded with the generated ontology because of the same reason described in CQ04, i.e. the absence of *ssn:MeasurementCapability* in SAREF and no mappings on the *ssn:hasMeasurementCapability* object property.

4.3 Discussion

The evaluation above demonstrated that from 5 competency questions only 3 (60%) could be answered with the mappings described in this paper. The main issue identified is the lack of a naive element in SAREF to describe the measurement capabilities of a sensor, which SSN enables through the *ssn:hasMeasurementCapability* object property of *ssn:Sensor*. To cope with this issue we suggest that a new mapping is created to align the structure from SSN, i.e. create the object property *ssn:hasMeasurementCapability* on *saref:Sensor* with the restriction of only *ssn:MeasurementCapability*. In addition, the mapping must consider to align both *ssn:MeasurementCapability* and *ssn:MeasurementProperty* as (is-a) *saref:Property*. This would enable the link of a *saref:Sensor* to the *ssn:MeasurementCapability*, which incorporates the links to the *ssn:MeasurementProperty*.

A conceptual issue in SSN was identified regarding the element *ssn:Sensor*. The description of this element states that it "allows sensors, methods, instruments, systems, algorithms and process chains as the process used of an observation (...) they are all grouped under the term sensor". Thus, the description includes that a *ssn:Sensor* can be a "system", but *ssn:Sensor* was not implemented as a specialization of *ssn:System*. In this way, *ssn:Sensor* could also inherit the composition relationship (*ssn:hasSubSystem*) and, thus, can represent a set of sensors. Issues identified in WM30 ontology include: (i) the *WM30:Vaisala_WM30* composition of wind direction (*WM30:WM30_WindDirection*) and speed (*WM30:WM30_WindSpeed*) sensors. Both are *ssn:Sensor*, but the composition relationship (*ssn:hasSubSystem*) is applied from a *ssn:System* to a *ssn:System*. Therefore, the M02 mapping must not consider the types of the subject or the object of the ssn:hasSubsystem property. (ii) the universal quantifier on *ssn:forProperty* (Figure 4) is wrong.

A practical issue when mapping to SAREF is to consider extending the taxonomy of sensor "types" by creating a new element when the type does not exist in SAREF. For example, smoke and temperature sensors are classified as *saref:SmokeSensor* and *saref:TemperatureSensor,* respectively, having function (*saref:hasFunction*) and measure property (*saref:measuresProperty*), linking the type of the sensor with functions it has and the type of property it measures (*saref:Smoke* and *saref:Temperature*. Thus, in our example, the correct implementation for the wind sensor is to create the subclass *saref:WindSensor*, with function sensing and measuring the properties of wind direction and wind speed, which is guaranteed by M01.

A limitation of this work is that this evaluation is a very preliminary application of the methodology, which is evaluated on a manual application of the algorithms on a single example. Furthermore, the use of pseudo-code to specify the translations seems not be the most adequate approach, which can be leveraged with a graphical modelling language for ontology alignments. Moreover, the number of competency questions(five) is too small and lacks on characterist-

ics represented in the original WM30 ontology. In future work it is intended to improve the evaluation process, so competency questions can be responded in different levels (e.g. fully, half, not answered). An issue that must be addressed is revisiting these mappings when SSN 2.0 is published, and decide whether the mappings will be updated or simply use the ontology alignments provided by SSN 2.0 specification. It includes the description of complex alignments which are presented with the property-chain axioms, as the alignment of *oldssn:observes* (property used in our third mapping) to the equivalent *sosa:observes* with chain axioms *oldssn:hasMeasurementCapability oldssn:forProperty* and *oldssn:madeObservation oldssn:observedProperty*. In either ways this work will still be applicable and will not become obsolete. Although this work only covers three vocabulary terms from the SSN 1.0, here we address the main elements used to characterize sensors, thus, providing a significant contribution for the state of the art.

5. CONCLUSIONS

In this paper we described the initial mappings between SSN and SAREF towards their ontological alignments to enable the semantic integration of IoT platforms relying on them. In particular, the mappings presented here focused in translating the main parts of a sensor ontology, i.e. the elements about sensor, device, its composition and functions. Here we detailed three mappings to cover these properties, showing how to produce a SAREF ontology from a SSN ontology through a semantic translation mechanism. An evaluation was performed to validate the initial specification produced, which used an ontology of wind sensor with SSN (WM30, provided by W3C) as input and generates a SAREF-based ontology as output. The results demonstrated a coverage of 60% of semantics maintained from the original ontology (SSN) to the generated (SAREF).

The main issue identified is the lack of measurement capabilities in SAREF to represent the collection of measurement properties of a component of a sensor. For example, the wind direction component of the WM30 wind sensor measures the wind direction property, which has a specific configuration of accuracy, delay distance, starting threshold and damping ratio. Therefore, we identified that the representation of these characteristics in SAREF needs to be addressed, probably by extending it with the *ssn:hasMeasurementCapability* object property (used by *saref:Sensor*) and importing *ssn:MeasurementCapability* and *ssn:MeasurementProperty* (as *saref:Property*).

From this initial iteration on the development of the bidirectional semantic translations we can conclude that, although the mappings presented here are quite intuitive, they reflect the foundations of the ontology alignments between SSN and SAREF and is a required first step towards complete semantic translations between them. Therefore, it represents a relevant contribution to the state-of-art by enabling a basic level of semantic interoperability between IoT platforms relying on SSN and SAREF.

Future work includes the acquisition and/or generation of test datasets (in both SSN and SAREF), giving emphasis to the requirements for an early warning system [11] to detect accidents in the port of Valencia, an INTER-IoT application scenario. New mappings will be created according to this scope with incremental evaluations, using the test datasets to verify whether the semantic interoperability is improved or not. Then, these mappings will be implemented through configurations in IPSM, exposing semantic translation services that will be consumed by the early warning system.

6. ACKNOWLEDGMENTS

Thanks for the CAPES funding agency (BEX 1046/14-4), TNO and EU-H2020-ICT grant INTER-IoT 687283.

7. REFERENCES

[1] M. Brambilla, J. Cabot, and M. Wimmer. *Model-Driven Software Engineering in Practice*. Morgan & Claypool Publishers, 1st edition, 2012.

[2] M. Compton, P. Barnaghi, L. Bermudez, and et al. The SSN ontology of the W3C semantic sensor network incubator group. *Journal of Web Semantics*, 17:25–32, 2012.

[3] M. Compton, H. Neuhaus, K. Taylor, and K. N. Tran. Reasoning about sensors and compositions. *CEUR Workshop Proceedings*, 522:33–48, 2009.

[4] L. Daniele, F. den Hartog, and J. Roes. Created in Close Interaction with the Industry: The Smart Appliances REFerence (SAREF) Ontology. In *7th international FOMI workshop*, volume 225, pages 100–112, 2015.

[5] L. Daniele, F. den Hartog, and J. Roes. How to Keep a Reference Ontology Relevant to the Industry: A Case Study from the Smart Home. *Lecture Notes in Computer Science (Artificial Intelligence)*, 9557:117–123, 2016.

[6] L. Daniele, M. Solanki, F. den Hartog, and J. Roes. *Interoperability for Smart Appliances in the IoT World*, pages 21–29. Springer International Publishing, Cham, 2016.

[7] M. Ganzha, M. Paprzycki, W. Pawłowski, and P. Szmeja. From implicit semantics towards ontologies — practical considerations from the INTER-IoT perspective. pages 59–64, 2017.

[8] M. Ganzha, M. Paprzycki, W. Pawlowski, P. Szmeja, and K. Wasielewska. Semantic technologies for the IoT - An Inter-IoT perspective. *Proceedings - 2016 IEEE 1st International Conference on Internet-of-Things Design and Implementation, IoTDI 2016*, pages 271–276, 2016.

[9] M. Ganzha, M. Paprzycki, W. Pawlowski, P. Szmeja, K. Wasielewska, and G. Fortino. Tools for Ontology Matching—Practical Considerations from INTER-IoT Perspective. 9258:143–154, 2015.

[10] M. Ganzha, M. Paprzycki, W. Pawłowski, P. Szmeja, and K. Wasielewska. Semantic interoperability in the Internet of Things: An overview from the INTER-IoT perspective. *Journal of Network and Computer Applications*, 2016.

[11] J. L. R. Moreira, L. Ferreira Pires, M. V. Sinderen, and P. D. Costa. Ontology-driven Conceptual Modeling for Early Warning Systems: Redesigning the Situation Modeling Language. In *MODELSWARD conference*, pages 467–477, 2017.

[12] J. Soldatos, N. Kefalakis, M. Hauswirth, M. Serrano, J.-P. Calbimonte, M. Riahi, K. Aberer, P. P. Jayaraman, A. Zaslavsky, I. P. Žarko, L. Skorin-Kapov, and R. Herzog. *OpenIoT: Open*

Source Internet-of-Things in the Cloud, pages 13–25.
Springer International Publishing, Cham, 2015.