Two Ontology Design Patterns toward Energy Efficiency in Buildings

Iker Esnaola-Gonzalez[1,2], Jesús Bermúdez[2], Izaskun Fernández[1], and Aitor Arnaiz[1]

[1] IK4-TEKNIKER, Iñaki Goenaga 5, 20600 Eibar, Spain
{iker.esnaola, izaskun.fernandez, aitor.arnaiz}@tekniker.es
[2] University of the Basque Country (UPV/EHU), Paseo Manuel Lardizabal 1, 20018 Donostia-San Sebastián, Spain
jesus.bermudez@ehu.eus

Abstract. Achieving an energy efficient operation of a building is not a straightforward task. In this article, two Ontology Design Patterns (ODP) are proposed, motivated by specific challenges that arise in this domain, and with the intention to support data analysts towards this goal. The two proposed ODPs are the AffectedBy ODP and the EEP (Execution-Executor-Procedure) ODP, which is an extension of the first. Both of them are intended to fill the gap that existing ontologies and ODPs fail to address adequately. Furthermore, both ODPs are aligned to an upper level ontology and some other related ontologies, which makes them applicable to other domains and scenarios.

1 Introduction

Buildings and construction account for more than 35% of global energy use and nearly 40% of energy-related CO2 emissions [1]. This is why efficient management of building energy plays a vital role and is becoming the trend for a future generation of buildings. Furthermore, since people spend more than 85% of their time in buildings [2], feeling comfortable while staying indoors is a must. In this context, the convergence of the Internet of Things' (IoT) rapid spread and the Knowledge Discovery in Databases (KDD) is expected to lead to significant progress. A KDD process can be understood as a five step process to extract useful knowledge from raw data and, in the energy efficiency field for buildings, they have traditionally been employed for tasks such as energy consumption forecasting [3]. However, having insufficient domain expertise can make data analysts feel overwhelmed throughout this process, resorting to a trial-and-error approach searching for variables and tasks to make accurate predictions. Consequently, the KDD process becomes arduous and time-consuming.

The EEPSA (Energy Efficiency Prediction Semantic Assistant) process addresses this problem taking leverage of Semantic Technologies like ontologies, ontology-driven rules and ontology-driven data access to guide data analysts through the different KDD phases in a semi-automatic manner, towards the enhancement of the KDD process [4]. In this process, the EEPSA ontology[3] plays a vital role capturing the necessary knowledge, mainly related to buildings, sensing and actuating devices, and their corresponding observations and actuations. This knowledge, along with other relevant domain and expert knowledge for the matter at hand, is captured in well-decoupled modules and represented in a form that can support data analysts.

In this paper, two Ontology Design Patterns (ODP) are proposed: the AffectedBy ODP and the Execution-Executor-Procedure (EEP) ODP, which is an extension of the AffectedBy ODP. Both ODPs are motivated by specific challenges that arise in problems related to energy efficiency in buildings, and they are defined with the intention to support data analysts throughout the KDD process. These two ODPs form the core of the renewed version of the EEPSA ontology. Furthermore, both ODPs are aligned to an upper level ontology and some other related ontologies, which makes them applicable to other domains and scenarios.

The rest of this paper is structured as follows. Section 2 introduces the related work. Section 3 describes the two ODPs. Finally, the conclusions of this work are presented in section 4.

2 Related Work

The energy efficiency in buildings domain spans concepts that overlap with the IoT field such as spaces, devices, observations, procedures, properties, and units of measurements to name a few. Some ontologies have considered these issues in their universe of discourse. However, the intended broad scope of these ontologies, usually cause large and complex bodies of terminology, and sometimes introduce too specific commitments that provoke a hard learning curve and hinder their reuse. An encouraging ontology design methodology to unlock these problems is the pattern-based ontology design. An ODP is a modelling solution to solve a recurrent ontology design problem [5]. Ideally, ODPs should be extendable but self-contained, minimize ontological commitments to foster reuse, address one or more explicit requirements (such as use cases or competency questions), be associatable to an ontology unit test, be the representation of a core notion in a domain of expertise, be alignable to other patterns, span more than one application area or domain, address a single invariant instead of targeting multiple reocurring issues at the same time, follow established modelling best practices, and so forth [6]. Following, a quick review of ODP-based ontologies related to sensing and actuating devices, and their context, is presented.

The DOLCE+DnS Ultralite (DUL[4]) ontology is a simplification of some parts of the DOLCE Lite-Plus library and Descriptions and Situations ontology. It is an upper-level ontology, and it defines general terms that are common across different domains. Therefore, it supports a broad semantic interoperability

[3] `https://w3id.org/eepsa`
[4] `http://www.ontologydesignpatterns.org/ont/dul/DUL.owl`

among domain-specific ontologies by providing a common starting point for the formulation of definitions.

The Semantic Sensor Network (SSN) ontology [7] was developed by the W3C Semantic Sensor Networks Incubator Group (SSN-XG[5]) and described sensors, observations and methods used for sensing among other concepts. It was aligned with the DUL ontology and built around a central ODP called Stimulus-Sensor-Observation [8] (SSO) describing the relationship between sensors, stimulus and observations. The W3C Spatial Data on the Web Working Group (SDWWG[6]) proposed an update of the SSN ontology[7] that became a W3C recommendation. The new version of the SSN ontology[8] follows a horizontal and vertical modularization architecture by including a lightweight but self-contained core ontology called SOSA[9] (Sensor, Observation, Sample, and Actuator) for its elementary classes and properties. Furthermore, similar to the original SSO patterns, SOSA acts as a central building block for the new SSN ontology. In line with the changes implemented for the new SSN ontology, SOSA also avoids the direct DUL import that previous version had, although an optional alignment can be achieved via the SSN-DUL alignment module[10].

The Actuation-Actuator-Effect[11] (AAE) ODP intends to model the relationship between an Actuator and the Effect it has on its environment through Actuations. This pattern adapts the SSN ontology's SSO ODP for actuators. The new version of the SSN ontology covers the function of the AAE ODP for actuators by expanding the SSO pattern in the SOSA ontology.

The SSN ontology does not provide enough constraints to the definitions of classes and properties to guarantee a proper answer to a question like: what is the feature of interest corresponding to a given property that has been observed by a sensor? And neither to this other question: which sensors observe a given property of a feature of interest? The patterns proposed in this paper solve these problems.

The SmartEnv ontology, proposed as a representational model to assist the development process of smart environments, is a network of 8 different ODPs [9]. These ODPs are used to modularize the proposed solution, while at the same time avoiding strong dependencies between the modules to manage the representational complexity of the ontology. The SmartEnv relies on the SSN ontology without introducing enough constraints to solve the aforementioned weaknesses of the SSN ontology.

The SEAS Ontology[10] is an ontology designed as a set of simple core ODPs that can be instantiated for multiple engineering related verticals. It is planned

[5] `https://www.w3.org/2005/Incubator/ssn/`

[6] `http://www.opengeospatial.org/projects/groups/sdwwg`

[7] `https://www.w3.org/TR/vocab-ssn/`

[8] `http://www.w3.org/ns/ssn/`

[9] `http://www.w3.org/ns/sosa/`

[10] `http://www.w3.org/ns/ssn/dul`

[11] `http://ontologydesignpatterns.org/wiki/Submissions:Actuation-Actuator-Effect`

to be added to the SAREF (Smart Appliances REFerence) ontology[12], which is expected to ease its adoption and extension by industrial stakeholders, while ensuring easy maintenance of its quality, coherence, and modularity [11]. The SEAS Feature of Interest ontology[13], is one of the modules that forms the SEAS ontology, and defines features of interest (*seas:FeatureOfInterest*) and properties (*seas:Property*). The Procedure Execution ontology[14] (PEP) defines procedure executors that implement procedure methods, and generate procedure execution activities. Furthermore, PEP defines an ODP as a generalization of SOSA's sensor-procedure-observation and actuator-procedure-actuation models. The patterns proposed in this paper are a reengineering of the PEP and the FeatureOfInterest ontologies and, additionally, an integration of them into a single ODP.

The Observation[15] ODP aims at representing observations of things, under a set of parameters. This set of parameters may include the place where the observation was made, the time when it was made, and any other feature concerning the specific thing being observed.

The IoT Application Profile (IoT-AP) ontology, is an ontology for representing and modelling the knowledge within the domain of the IoT [12]. The ontology is designed re-using ODPs such as the aforementioned Observation ODP. It focuses in observations, but it also covers sensors that make those observations, values of those observations and observation collections. However, this ontology suffers from similar weaknesses to those previously commented about the SSN ontology. This is basically due to the lack of proper constraints on property definitions.

The ODP repository[16] collects and makes ODPs available on the web, allowing users to download, propose, and discuss them. Some of the mentioned ODPs are hosted in this repository.

3 Motivation and Pattern Overview

The EEPSA ontology supports data analysts that are not experts in the energy efficiency in tertiary buildings domain, towards the creation of enhanced predictive models. For that purpose, the ontology not only needs to contain both domain knowledge and expert knowledge, but also needs to represent it in a way that can be leveraged to guide data analysts throughout the KDD process.

In energy efficiency problems related to tertiary buildings domain, two recurrent modelling challenges arise. The first one is related to modelling variables that may affect an indoor condition such as indoor temperature or occupancy. The second one is related to modelling the variables measured within a building and the systems to measure them. The definition of ODPs for these problems

[12] https://w3id.org/saref
[13] `https://ci.mines-stetienne.fr/seas/FeatureOfInterestOntology`
[14] `https://ci.mines-stetienne.fr/pep/`
[15] `http://ontologydesignpatterns.org/wiki/Submissions:Observation`
[16] `http://www.ontologydesignpatterns.org`

would be beneficial and could ideally act as building blocks to be reused in case someone else faces these same modelling challenges.

3.1 AffectedBy

In the first phase of a typical KDD process, known as the Data Selection phase, the EEPSA process supports data analysts selecting datasets and subset of variables or data samples that are relevant for the matter at hand. Taking into account that data analysts may not be experts in the energy efficiency in tertiary buildings domain, they may feel overwhelmed during this task, due to their lack of expertise in choosing the adequate variables. Therefore, they would benefit from a resource that supports the discovery of relevant variables that affect the environment of a given space or another feature of interest. Any of these variables will be represented as properties or qualities of a feature of interest.

For example, let us consider the LR03 lecture room as a feature of interest: a lecture room located on the ground floor of a building, and with a large window that overlooks the road that passes near the building. Some properties of LR03 are: the area of the lecture room, the number of seats available or the quality of comfort at any given time. The quality of comfort in this lecture room is affected by the room temperature and the nearby outdoor noise. In turn, the temperature of LR03 is affected by the number of people present in the lecture room, the humidity of the lecture room, and the intensity of solar radiation received through the room window. Regarding the impact of outdoor noise, it is affected by the sound insulation factor of the room. Lastly, the received solar radiation is affected by the azimuth (i.e. orientation) of the lecture room window.

The following competency questions must be considered:

- CQ1: What are the properties/qualities of a feature of interest?
- CQ2: What are the properties/qualities that affect a given property of a feature of interest?
- CQ3: Which feature of interest does a given property/quality belongs to?

The SSN Ontology contains a building block that may be useful for this matter. However, an inadequacy was spotted. The *ssn:Property* class is textually defined as “a quality of an entity. An aspect of an entity that is intrinsic to and cannot exist without the entity”. This definition is made basically, according to the definition of the *dul:Quality* class. In fact, it is declared[17] that *ssn:Property rdfs:subClassOf dul:Quality*. Furthermore, the *ssn:Property* class is linked to the *ssn:FeatureOfInterest* class with the *ssn:isPropertyOf* object property. Nevertheless, this object property is not functional, meaning that the *ssn:isPropertyOf* property can have more than one value for the same individual, so the following triples can be found in a ssn-annotated triple set:

```
:temperature rdf:type ssn:Property.
:temperature ssn:isPropertyOf :lr03.
```

[17] https://www.w3.org/TR/vocab-ssn

```
:lr03 rdf:type ssn:FeatureOfInterest.

:temperature ssn:isPropertyOf :lr07.
:lr07 rdf:type ssn:FeatureOfInterest.

:lr03 owl:differentFrom :lr07.
```

According to the aforementioned *ssn:Property*'s class textual definition, individual *:temperature* is intrinsic to and cannot exist without the existence of individual *:lr03*. However, the triples shown contradict such definition (i.e., *:temperature* is a quality of different entities). Probably, designers of the SSN ontology would advise against this practice and, in fact, example "B.3 apartment 134"[18] uses the URI <apartment/134/electricConsumption> for referring to an individual that represents the electrical consumption of the apartment #134. However, the identification of the feature of interest (i.e., apartment #134) of this property is embedded in the URI and this is not enough for machine interpretation. Of course, two different rooms may have the same temperature value (e.g. 15°C) but such circumstance would be represented as a property value of each different temperature instances. Moreover, a suitable hierarchy of *Property* subclasses may be desirable. A class *Temperature* would be a subclass of class *Property*. Therefore, it could be possible to ask for all the features of interest that have a temperature quality.

The issue mentioned above is tackled in the SEAS Feature of Interest ontology, where an ODP to describe features of interest and their properties is defined. In this pattern, the *seas:isPropertyOf* object property links a *seas:Property* to a *seas:FeatureOfInterest*, and it is declared as subproperty of *ssn:isPropertyOf*. However, *seas:isPropertyOf* is functional. Therefore, it represents more faithfully the textual definition of *ssn:Property*.

Furthermore, the SEAS Feature of Interest ontology also defines the *seas:derivesFrom* object property which links a *seas:Property* to another *seas:Property* it derives from. This object property is defined as a symmetric property, which means that the property has itself as inverse. However, this constraint is unnecessary and sometimes even inappropriate. For instance, the temperature of individual *:lr03* may derive from the occupancy of the room, but the occupancy does not necessarily derive from the temperature of the room.

In addition, the SEAS Feature of Interest ontology contains a textual comment that, although relevant, it is not materialized as an axiom:

$$\textit{seas:hasProperty} < \textit{seas:hasProperty} \circ \textit{seas:derivesFrom}$$

The AffectedBy ODP is inspired by the identified SSN ontology and SEAS Feature of Interest ontology weaknesses. It defines the building block shown in Figure 1, that consists of two classes: *aff:FeatureOfInterest* and *aff:Quality*, and three properties: *aff:hasQuality*, *aff:belongsTo*, and *aff:affectedBy*.

[18] `https://www.w3.org/TR/vocab-ssn/#apartment-134`

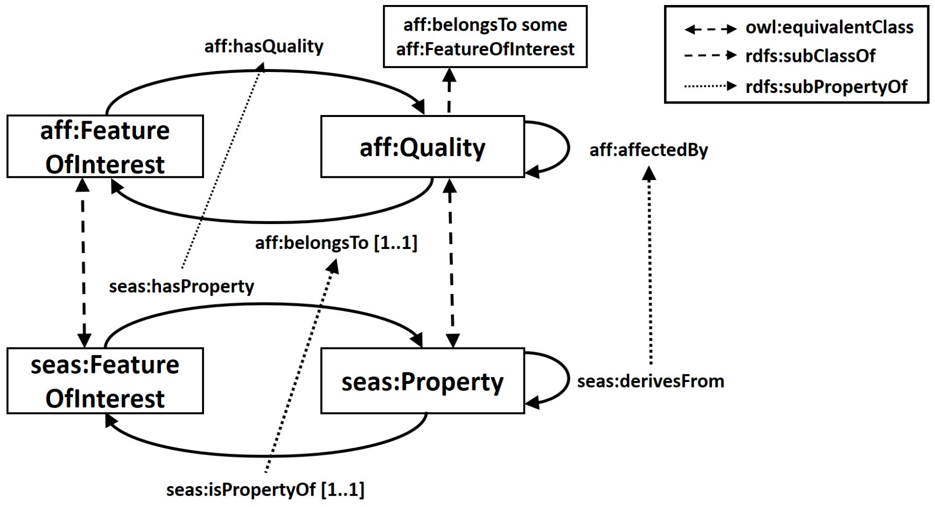

Fig. 1. The AffectedBy ODP.

The property *aff:affectedBy* (released from the symmetric constraint) is defined in the AffectedBy ODP to replace the role of the property *seas:derivesFrom*. It can be asserted that *seas:derivesFrom* is a subproperty of *aff:affectedBy*. The class *aff:FeatureOfInterest* is equivalent to *seas:FeatureOfInterest*, and the class *seas:Property* is equivalent to *aff:Quality*. Moreover, *seas:hasProperty* is subproperty of *aff:hasQuality*, and *seas:isPropertyOf* is subproperty of *aff:belongsTo*. Furthermore, *aff:belongsTo* is defined to be functional and it is the inverse of *aff:hasQuality*, to support the notion that a quality is intrinsic to the feature of interest (i.e., an entity) to which it belongs (according to the conceptualization in DUL); and it is also asserted that every quality belongs to a feature of interest, i.e.,

aff:Quality rdfs:subClassOf aff:belongsTo some aff:FeatureOfInterest).

Finally, the following property chain axiom is asserted:

aff:hasQuality ∘ *aff:affectedBy rdfs:subPropertyOf aff:hasQuality*

Even though the ODP is motivated by the energy efficiency in buildings problem, it is applicable to similar problems from different domains. Therefore, the AffectedBy ODP is aligned with the DUL ontology. Moreover, the AffectedBy ODP is also aligned to the SSN Ontology and the SEAS Feature of Interest ontology. The alignments with these three ontologies are kept in separate files[19].

[19] `https://github.com/iesnaola/AffectedBy/tree/master/alignments`

Likewise, the HTML documentation of the ODP is available[20] via LODE (Live OWL Documentation Environment [13]) and in the ODP repository[21].

Application. The instantiation of the AffectedBy ODP for the aforementioned LR03 lecture room is shown in Figure 2. For the sake of simplicity, the *rdf:type* relationships are not shown.

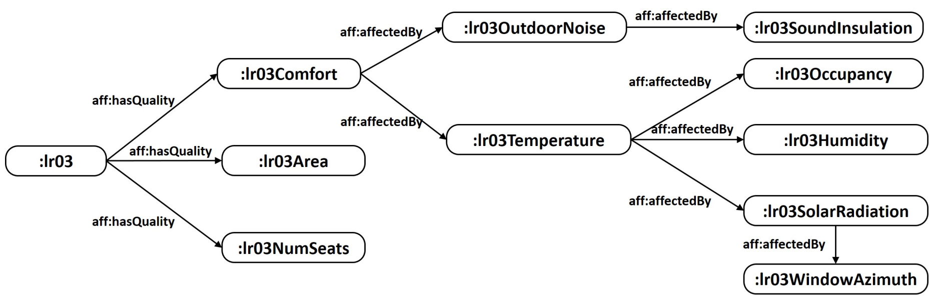

Fig. 2. The AffectedBy implementation in the LR03 Lecture Room.

With respect to this example, the following competency questions can be applied and answered:

– (CQ2): What are the properties that affect the property *:lr03Comfort*?
SPARQL query: SELECT *?x* WHERE {*:lr03Comfort aff:affectedBy ?x.*}
Answer: *:lr03Temperature*, *:lr03OutdoorNoise.*

– (CQ2): What are the properties that affect the property *:lr03Temperature*?
SPARQL query: SELECT *?x* WHERE {*:lr03Temperature aff:affectedBy ?x.*}
Answer: *:lr03Occupancy*, *:lr03Humidity*, *:lr03SolarRadiation.*

– (CQ1): What are the properties of the feature of interest *:lr03*?
SPARQL query: SELECT *?x* WHERE {*:lr03 aff:hasQuality ?x.*}
Answer: *:lr03Area*, *:lr03NumSeats :lr03Comfort*, *:lr03Temperature*, *:lr03OutdoorNoise*, *:lr03Occupancy*, *:lr03Humidity*, *:lr03SolarRadiation*, *:lr03SoundInsulation*, *:lr03WindowAzimuth.*
(After inferences provided by the axiom *aff:hasQuality* $\circ$ *aff:affectedBy rdfs:subPropertyOf aff:hasQuality*).

– (CQ3): Which feature of interest does the property *:lr03SolarRadiation* belongs to?
SPARQL query: SELECT *?x* WHERE {*:lr03SolarRadiation aff:belongsTo*

[20] `https://w3id.org/affectedBy`
[21] `http://ontologydesignpatterns.org/wiki/Submissions:AffectedBy`

?x.}
Answer: *:lr03.*
(After inferences provided by the axioms *aff:hasQuality* ∘ *aff:affectedBy* *rdfs:subPropertyOf* *aff:hasQuality* and *aff:belongsTo* *inverseOf* *aff:hasQuality*).

3.2 Execution-Executor-Procedure (EEP)

An interesting information for data analysts could be: which are the sensors/actuators deployed in the space where the energy efficiency is aimed? And even more: which are the capabilities of those sensors/actuators? Moreover, knowing this information would let data analysts make further queries to discover sensors or actuators that observe or act on a given property of a space. More specifically, the CQs considered are the following:

- CQ1: What are the observations/actuations performed by a given procedure?
- CQ2: What are the observations/actuations performed by a given sensor/actuator?
- CQ3: What are the procedures implemented by a given sensor/actuator?
- CQ4: What are the features of interest on a given observation/actuation?
- CQ5: What are the properties/qualities sensed/actuated by a given observations/actuations?
- CQ6: What are the features of interest of a given sensor/actuator?
- CQ7: What are the properties/qualities sensed/actuated by a given executor?

For each competency question CQn, we can consider a twin competency question CQn^{i} which consists on rephrasing the question in the opposite direction. For instance, CQ1^{i} is defined as "What is the procedure used in a given observation/actuation?". In terms of a SPARQL query, it means that the query variable is moved from the subject position to the object position, or the other way round, of the triple pattern.

These questions have been tackled by the SSN ontology with SOSA's Observation-Sensor-Procedure pattern, and by the SAN ontology with the AAE pattern. However, in their current state, they cannot properly fulfil the discovery of sensors and actuators because no property has been defined that directly links sensors or actuators to features of interest, and moreover, compositions of properties that link them through the Observation or Actuation class, are not sufficiently constrained to satisfy the aforementioned competency questions. For instance, the following set of ssn-annotated triples is not enough to answer the question: which is the sensor that observes the temperature of :lr07?

```
:sensor1 sosa:madeObservation :obs1;
         sosa:observes :temperature.
:temperature ssn:isPropertyOf :lr03.
:obs1 sosa:hasFeatureOfInterest :lr03.
```

```
:sensor2 sosa:madeObservation :obs2;
         sosa:observes :temperature.
:temperature ssn:isPropertyOf :lr07.
:obs2 sosa:hasFeatureOfInterest :lr07.
:sensor1 sosa:madeObservation :obs3;
         sosa:observes :humidity.
:humidity ssn:isPropertyOf :lr07.
:obs3 sosa:hasFeatureOfInterest :lr07.
```

In order to fill this gap, the Execution-Executor-Procedure (EEP) ODP presented in this paper represents executions (e.g., events such as observations or actuations) made by executors (e.g., systems such as sensors or actuators) that implement procedures to carry out their goals. Executions and executors are taken over features of interest and their intrinsic properties or qualities.

The EEP ODP is an adaptation of the PEP ontology from the SEAS ontology which, in turn, is a generalization of the Observation-Sensor-Procedure and Actuation-Actuator-Procedure patterns used in the SOSA and SSN ontologies. The EEP ODP imports the AffectedBy ODP that involves classes for features of interest and their intrinsic properties/qualities. Furthermore, from the AffectedBy ODP, the EEP ODP imports the notion that a property/quality is intrinsic to the feature of interest that it belongs to (i.e., according to the definition of the class Quality in the DUL ontology).

Apart from the two classes (i.e., *aff:FeatureOfInterest* and *aff:Quality*) imported from the AffectedBy ODP, the EEP ODP consists of three more classes: *eep:Execution*, *eep:Executor*, and *eep:Procedure* (see Figure 3). An individual of *eep:Execution* is an action related to a property of a feature of interest, produced by an agent by performing a procedure. An individual of *eep:Executor* is an agent capable of performing tasks by following procedures. An individual of *eep:Procedure* is a description of some actions to be executed by agents. The class *eep:Execution* and their three functional object properties *eep:madeBy*, *eep:usedProcedure*, and *eep:onQuality*, form the backbone of the ODP. The property *eep:madeBy* links an execution to the agent that performs the action; the property *eep:usedProcedure* links an execution to the procedure that describes the task to be performed; and the property *eep:onQuality* links an execution to the quality/property concerned by the execution. Therefore, an execution jointly with their three object values of the three aforementioned properties can be considered as a n-ary relationship. Note that every quality belongs to a unique feature of interest, so a feature of interest is also involved in the n-ary relationship.

The remaining object properties are: *eep:implements*, linking executors to procedures; *eep:hasFeatureOfInterest*, linking executions to features of interest; *eep:forQuality*, linking executors to qualities; and *eep:forFeatureOfInterest*, linking executors to features of interest. Note that an executor can implement different procedures corresponding to different executions performed by the same executor. Analogously, an executor may be committed to different properties and features of interest. These four properties are defined in terms of the functional

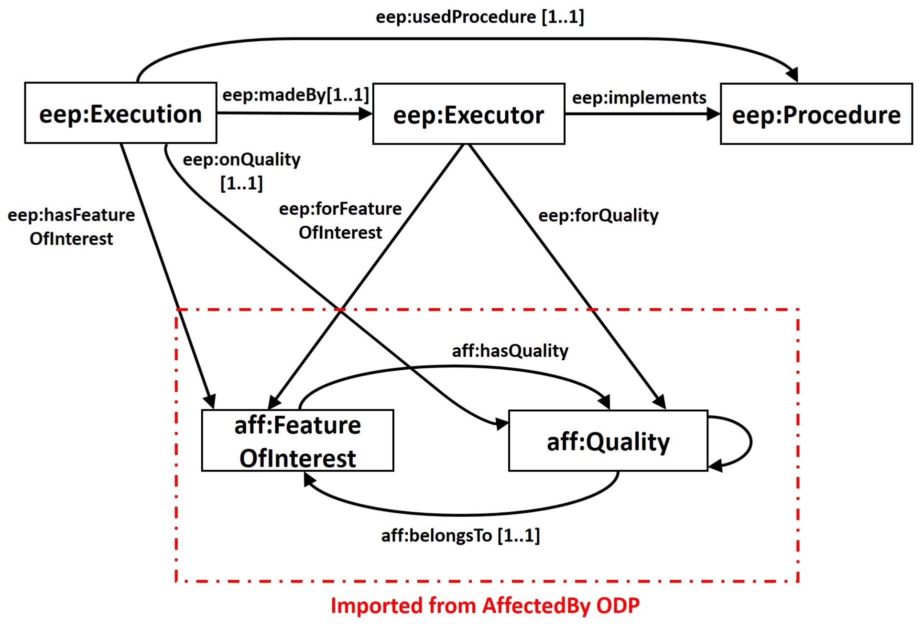

Fig. 3. The Execution-Executor-Procedure (EEP) ODP.

object properties using the following property chain axioms:

inverse(eep:madeBy) ∘ *eep:usedProcedure rdfs:subPropertyOf eep:implements.*
eep:onQuality ∘ *eep:belongsTo rdfs:subPropertyOf eep:hasFeatureOfInterest.*
inverse(eep:madeBy) ∘ *eep:onQuality rdfs:subPropertyOf eep:forQuality.*
eep:forQuality ∘ *eep:belongsTo rdfs:subPropertyOf eep:forFeatureOfInterest.*

Axioms included in the EEP ODP provide inferences that allow to answer the formulated CQs properly, solving the previously referred weaknesses of the sosa/ssn ontologies. Note that only triples about the four functional object properties *eep:madeBy*, *eep:usedProcedure*, *eep:onQuality*, and *aff:belongsTo*, need to be asserted, and the remaining triples are inferred by the property axioms.

Likewise the AffectedBy ODP, the EEP ODP is motivated by the energy efficiency in buildings problem but it is applicable to different domains. It is aligned with the DUL ontology, the SSN Ontology, and the PEP ontology. The alignments with these three ontologies are kept in separate files[22]. Furthermore, the HTML documentation of the ODP is available[23] via LODE and in the ODP repository[24].

[22] https://github.com/iesnaola/EEP/tree/master/alignments
[23] https://w3id.org/eep
[24] http://ontologydesignpatterns.org/wiki/Submissions:EEP

Application. The EEP ODP is instantiated in a farm scenario where poultry are reared. In this case, a sensor *:sensor36* deployed in the farm individual *:farm* is in charge of measuring both farm's temperature and humidity (i.e., *:farmTemperature* and *:farmHumidity*). Furthermore, this sensor implements a monitoring procedure (*:monitoringProc*) to make two observations *:obs13* and *:obs14*. Figure 4 shows this instantiation.

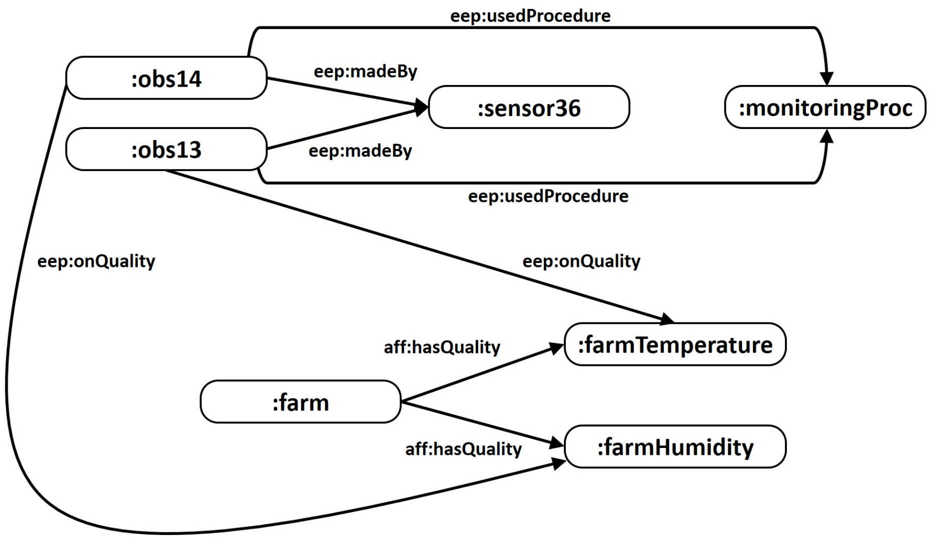

Fig. 4. The Execution-Executor-Procedure (EEP) ODP implementation in a farm.

With respect to this example, the following competency questions can be applied and answered:

– (CQ1): What are the executions performed by procedure *:monitoringProc*?
SPARQL query: SELECT *?x* WHERE {*?x eep:usedProcedure :monitoring-Proc.*}
Answer: *:obs13*, *:obs14*.

– (CQ2): What are the observations performed by sensor *:sensor36*?
SPARQL query: SELECT *?x* WHERE {*?x eep:madeBy :sensor36.*}
Answer: *:obs13*, *:obs14*.

– (CQ3): Which are the procedures implemented by the sensor *:sensor36*?
SPARQL query: SELECT *?x* WHERE {*:sensor36 eep:implements ?x.*}
Answer: *::monitoringProc*
(After inferences provided by the axiom *inverse(eep:madeBy)* ∘ *eep:used-Procedure rdfs:subPropertyOf eep:implements*).

- (CQ4$^{\mathrm{i}}$): What are the executions on the feature of interest *:farm*?
  SPARQL query: SELECT *?x* WHERE {*?x eep:hasFeatureOfInterest :farm.*}
  Answer: *:obs13*, *:obs14*.
  (After inferences provided by the axioms *eep:onQuality* ∘ *eep:belongsTo rdfs:subPropertyOf eep:hasFeatureOfInterest* and *aff:belongsTo inverseOf aff:hasQuality*).

- (CQ5): What are the qualities observed by the observation *:obs13*?
  SPARQL query: SELECT *?x* WHERE {*:obs13 eep:onQuality ?x.*}
  Answer: *:farmTemperature.*

- (CQ6$^{\mathrm{i}}$): What are the executors that observe/act on the feature of interest *:farm*?
  SPARQL query: SELECT *?x* WHERE {*?x eep:forFeatureOfInterest :farm.*}
  Answer: *:sensor36.*
  (After inferences provided by the axioms *eep:forQuality* ∘ *eep:belongsTo rdfs:subPropertyOf eep:forFeatureOfInterest* and *inverse(eep:madeBy)* ∘ *eep:onQuality rdfs:subPropertyOf eep:forQuality*).

- (CQ7): What are the qualities observed by sensor *:sensor36*?
  SPARQL query: SELECT *?x* WHERE {*:sensor36 eep:forQuality ?x.*}
  Answer: *:farmTemperature*, *:farmHumidity.*
  (After inferences provided by the axiom *inverse(eep:madeBy)* ∘ *eep:onQuality rdfs:subPropertyOf eep:forQuality*).

The EEP ODP presented in this paper left out the temporal context, which undoubtedly is a relevant issue. In fact, EEP can be easily extended by importing different conceptualizations of such temporal aspect. For instance, a simple solution is to define a property like *:atTime* linking *eep:Execution* to *:TimeInterval* (like in IoT-AP ontology, or similarly in Fiesta-IoT ontology [14]), and additionally to include some other property like *sosa:resultTime* linking *eep:Execution* to *xsd:dateTime* in order to differentiate the temporal entity that applies to the execution from the instant the execution was completed (as it is made in SOSA ontology). Moreover, a more complex conceptualization may be necessary in a scenario where executions are also features of interest and time is a quality of these executions, then *:Time* may be a subclass of *aff:Quality* (similarly to what is done in SAREF ontology). Otherwise, features of interest and their properties may need to be qualified by time-related properties; for instance, state duration of a feature of interest or change frequency of a property during a temporal context (as it is proposed in the SEAS Time Ontology[25]). In summary, EEP is ready to incorporate the preferred solution adopted by the EEP user.

[25] https://ci.mines-stetienne.fr/seas/TimeOntology

4 Conclusions and Future Work

In this paper we presented two ODPs: the AffectedBy ODP and the EEP (Execution-Executor-Procedure) ODP, which is an extension of the first. Both of them are expected to solve recurrent design problems that arise in energy efficiency problems for buildings and that are not adequately addressed by existing ontologies and ODPs. Both ODPs are aligned with related ontologies which make them applicable to similar problems in different domains, and are stored in the ODP repository.

In future work, both ODPs are expected to be the base for building ontology modules. Namely, they are planned to be the foundation for the reengineering of the measurements4EEPSA ontology module (an adapted extraction of the m3-lite[26] and QUDT[27] ontologies) responsible for containing measurements and device related knowledge, and a new ontology module containing expert knowledge in the energetic field.

Furthermore, these ODPs will be used in KDD processes with other goals that differ from attempting an energy efficient management of a building. Namely, they are expected to support the energy production forecasting of a Photovoltaic (PV) system as well as the management of Demand-Response strategies.

Acknowledgement

Part of the presented work received funding from FEDER/TIN2016-78011-C4-2-R. This work was conducted using the Protégé resource, which is supported by grant GM10331601 from the National Institute of General Medical Sciences of the United States National Institutes of Health.

References

1. T. Abergel, B. Dean and J. Dulac, Towards a zero-emission, efficient, and resilient buildings and construction sector. Global Status Report 2017, Technical Report, 2017. ISBN 978-92-807-3686-1. `http://www.worldgbc.org/sites/default/files/UNEP%20188_GABC_en%20%28web%29.pdf`.
2. N.E. Klepeis, W.C. Nelson, W.R. Ott, J.P. Robinson, A.M. Tsang, P. Switzer, J.V. Behar, S.C. Hern and W.H. Engelmann, The National Human Activity Pattern Survey (NHAPS): a resource for assessing exposure to environmental pollutants, *Journal of Exposure Science and Environmental Epidemiology* **11**(3) (2001), 231.
3. A.S. Ahmad, M.Y. Hassan, M.P. Abdullah, H.A. Rahman, F. Hussin, H. Abdullah and R. Saidur, A review on applications of ANN and SVM for building electrical energy consumption forecasting, *Renewable and Sustainable Energy Reviews* **33** (2014), 102–109, ISSN 1364-0321. doi:https://doi.org/10.1016/j.rser.2014.01.069. `http://www.sciencedirect.com/science/article/pii/S1364032114000914`.

[26] http://ontology.fiesta-iot.eu/ontologyDocs/m3-lite.owl
[27] http://www.qudt.org

4. I. Esnaola-Gonzalez, J. Bermúdez, I. Fernandez and A. Arnaiz, Semantic Prediction Assistant Approach applied to Energy Efficiency in Tertiary Buildings, *Semantic Web* (to appear). `http://www.semantic-web-journal.net/`.
5. A. Gangemi and V. Presutti, Ontology Design Patterns, in *Handbook on Ontologies*, S. Staab and R. Studer, eds, Springer Berlin Heidelberg, Berlin, Heidelberg, 2009, pp. 221–243. ISBN ISBN 978-3-540-92673-3. doi:10.1007/978-3-540-92673-3_10.
6. P. Hitzler, A. Gangemi and K. Janowicz, *Ontology Engineering with Ontology Design Patterns: Foundations and Applications*, Vol. 25, IOS Press, 2016.
7. M. Compton, P. Barnaghi, L. Bermudez, R. García-Castro, O. Corcho, S. Cox, J. Graybeal, M. Hauswirth, C. Henson and A. Herzog, The SSN ontology of the W3C semantic sensor network incubator group, *Web Semantics: Science, Services and Agents on the World Wide Web* **17** (2012), 25–32. doi:https://doi.org/10.1016/j.websem.2012.05.003.
8. K. Janowicz and M. Compton, The Stimulus-Sensor-Observation Ontology Design Pattern and its Integration into the Semantic Sensor Network Ontology., in: *SSN*, 2010.
9. M. Alirezaie, K. Hammar and E. Blomqvist, SmartEnv as a Network of Ontology Patterns, *Semantic Web* (to appear). `http://www.semantic-web-journal.net/`.
10. M. Lefranois, Planned ETSI SAREF Extensions based on the W3C&OGC SOSA/SSN-compatible SEAS Ontology Patterns, in: *Workshop on Semantic Interoperability and Standardization in the IoT, SIS-IoT*, 2017, p. 11.
11. L. Daniele, F. den Hartog and J. Roes, Created in close interaction with the industry: the smart appliances reference (SAREF) ontology, in: *International Workshop Formal Ontologies Meet Industries*, Springer, 2015, pp. 100–112. doi:https://doi.org/10.1007/978-3-319-21545-7_9.
12. A. Gangemi, R. Lillo, G. Lodi and A.G. Nuzzolese, A pattern-based ontology for the Internet of Things, *Proceedings of the 8th Workshop on Ontology Design and Patterns (WOP 2017)* **2043** (2017), ISSN 1613-0073. `http://ceur-ws.org/Vol-2043/paper-11.pdf`.
13. S. Peroni, D. Shotton and F. Vitali, The Live OWL Documentation Environment: a tool for the automatic generation of ontology documentation, in: *International Conference on Knowledge Engineering and Knowledge Management*, Springer, 2012, pp. 398–412.
14. R. Agarwal, D.G. Fernandez, T. Elsaleh, A. Gyrard, J. Lanza, L. Sanchez, N. Georgantas and V. Issarny, Unified IoT Ontology to Enable Interoperability and Federation of Testbeds, in: *3rd IEEE World Forum on Internet of Things*, 2016. doi:10.1109/WF-IoT.2016.7845470.