Semantic Web 1 (0) 1–5
IOS Press

SmartEnv as a Network of Ontology Patterns

Marjan Alirezaie [a,*], Karl Hammar [b], Eva Blomqvist [c]

[a] *Center for Applied Autonomous Sensor Systems, Örebro University, Sweden* - marjan.alirezaie@oru.se
[b] *SICS - East Swedish ICT, Linköping, Sweden, and Jönköping University, Sweden* - karl@karlhammar.com
[c] *SICS - East Swedish ICT, Linköping, Sweden* - eva.blomqvist@liu.se

Editor: Aldo Gangemi, ISTC-CNR Roma, Italy
Solicited reviews: Marta Sabou, MODUL University Vienna, Austria; Maxime Lefrançois, École des Mines de Saint-Étienne, France; Valentina Presutti, ISTC-CNR Roma, Italy

Abstract. In this article we outline the details of an ontology, called SmartEnv, proposed as a representational model to assist the development process of smart (i.e., sensorized) environments. The SmartEnv ontology is described in terms of its modules representing different aspects including physical and conceptual aspects of a smart environment. We propose the use of the Ontology Design Pattern (ODP) paradigm in order to modularize our proposed solution, while at the same time avoiding strong dependencies between the modules in order to manage the representational complexity of the ontology. The ODP paradigm and related methodologies enable incremental construction of ontologies by first creating and then linking small modules. Most modules (patterns) of the SmartEnv ontology are inspired by, and aligned with, the Semantic Sensor Network (SSN) ontology, however with extra interlinks to provide further precision and cover more representational aspects.

The result is a network of 8 ontology patterns together forming a generic representation for a smart environment. The patterns have been submitted to the ODP portal and are available on-line at stable URIs.

Keywords: Smart Environments, Ontology Design Pattern, Semantic Sensor Network

1. Introduction

Applications of sensorized environments that provide domestic monitoring and cognitive assistance services for their inhabitants/users are increasing. An example of such an application is health care monitoring and services, where patients are being monitored and cared for in their own living environment. In order to support the use of artificial intelligence techniques for automating the provision of these services, it is necessary to describe the capabilities of the various aspects of such environments. These descriptions include physical aspects (e.g., the structure of the environment, sensor network setting or entities), as well as conceptual aspects (e.g., events or activities of the users), and can be modeled in ontologies. According to [1], there is a general list of questions about the inhabitants of smart homes (as an example of smart environments), which many of the suggested knowledge models in the literature aim to address. This list includes questions about the location of the inhabitant, the activity that the inhabitant carries out, the intention behind the activity, the time when the activity is detected, etc. Although the representational models (i.e., ontologies) target the same goal, they differ in terms of the level of generality as well as their reasoning efficiency.

Due to the dependencies between the aforementioned features, such an ontology can easily become large and complex; moreover, it may need to be updated over time, e.g., when sensors/robots with new kinds of features are added to (or removed from) the environment, or when the monitoring requirements of the environment change. Additionally, when we use ontologies in a system that requires near real-time reasoning and reactions by the system, the reasoning complexity is an essential parameter of the ontologies. For these reasons, we propose

*

the use of a network of ontology modules, which is called SmartEnv and could be considered as a set of interlinked content Ontology Design Patterns (ODPs) [15] due to their generic applicability to the domain and our deliberate effort to minimize their ontological commitments, while maintaining functionality. According to the principles of ODP modeling, the ontological commitments should be minimized by first creating small modules, and then linking them together, instead of designing a large monolithic ontology from scratch as a comprehensive representation of the entire domain [16].

Before going to the details of our proposed ontology, in section 2, we first motivate how and why our suggested SmartEnv ontology is required and compare the available ontologies with the given requirement list. In section 3, we shortly introduce the project whose requirements led us to the development of the SmartEnv ontology in the form of a network of ontology modules. In section 4, we explain the 8 ontology patterns (modules) used in SmartEnv ontology. The SmartEnv ontology, as the result of specializing the relations between modules along with its application is presented in section 5. A discussion on our approach is given in section 6, where the paper is concluded by giving remarks concerning future developments.

2. Motivation and Background

2.1. Representational Requirements

During the requirements analysis process, we considered a number of (conceptual) aspects of smart environments to be covered in the ontologies. For further clarification, some aspects are explained based on a set of high-level competency questions (CQs) (only examples are given here due to space restrictions). The detailed CQs for each of the patterns can be found inside the ODP itself, as annotations of its OWL file, and on its page in the ODP portal:

Observation/Sensing Observing (i.e., monitoring) of an object or a place is the main motivation why the environment is sensorized. A representation model is required to answer questions such as what can be observed by a certain sensor? To what object is a sensor attached? What is the location of the object, and what does the sensor measure? Can the sensor or its holding object move?

Agents Agents (e.g, inhabitants of a home) are the main characters whose activities, locations, or more specific parameters such as safety and health are usually the main reason behind any observation process in a smart environment. A representation model is required to answer questions such as what are the possible activities of the agent? Can the agent be targeted by sensors? Where is the agent now? What is the agent doing now?

Activities/Events Any changes in a smart environment are represented in the form of an event or an activity. Questions such as when an event has occurred, or why such event was recognized, can be answered by representing activities in terms of their preconditions.

Objects Physical objects are also directly or indirectly the target of the observation process in order to recognize activities in a smart environment. We represent objects to answer questions about their state (being in a specific situation), locations, or the events or activities in which they are involved.

Network set-up In order to set-up a smart environment a sensor network deployment related to the observation process, is indispensable. A network representation model is used to answer any question regarding the hardware and software configuration of a network, its components and their locations.

Spatial aspect Any physical entity such as objects, agents, and places in a smart environment has a geometrical aspect based on which their spatial relations with the environment can be represented.

Temporal aspect Similar to spatial aspects, the temporal aspects are the main basis of an observation process. A temporal representation model is used to answer questions such as when the occurrence of an activity is realized. It also allows us to define activities based on the temporal relations with their preconditions.

Given the aforementioned general list of high level concepts required to represent a smart environment, in the following, we overview the literature on the sensor-related ontologies.

2.2. Overview of Sensor-Related Ontologies

Sensing and in general sensor-related details of smart environments as one of their integral computational aspects has been widely studied in the

literature [1, 2, 18]. Although the "re-usability" has always been advertised as an essential features of ontologies, there is a large number of adhocly designed sensor-related ontologies in the literature, many of which are not even available online. This number increases when the idea behind the design of these ontologies is getting more specific, for example for activity recognition purposes [1]. One important task in designing ontologies for smart environments is to identify the activities of an agent in the environment, for instance in the case of a smart home, these could be activities such as "sleeping", "watching TV", "cooking" etc. In addition to being able to identify an activity, we also need to represent the time and location where the activity is carried out. Ontologies suggested for recognizing daily activities are rarely relying on upper-level ontologies [19, 20], which usually result in a lower reusability. Concerning the re-usability of ontologies, [3] proposes a top-level ontology that provides a formal and generic representation of activities sharable between different domains. However, apart from the activity representation, in the aforementioned work there is no representation of the other aspects of a smart environment. With the focus on smart environments, [4] introduces the Casas Ontology (COSE) being able to represent sensors, buildings, occupants and human activities, which is publicly available. However, there are no representation details provided to show that the offered light-weight model relies on available upper-level ontologies, or even that it is possible to align it to such ontologies. Furthermore, both the spatial and temporal aspects are poorly represented and there is no support for the representation of the sensor network at all. Likewise, the COBRA-ONT ontology, which also provides a representational model for pervasive computing environments, lacks an alignment with an upper level ontology, and does not contain an explicit temporal representation model [5]. Two other ontologies related to context-awareness in sensorized environments are SOUPA [7] and DogOnt [8]. Although there are a number of working applications of these ontologies, both are lacking some representational aspects. For instance, in SOUPA, we are not able to define devices (as part of the network modules) and their configurations and functionalities. DogOnt also provides a limited object representation with no support for their dynamic features. Comparing with the above mentioned ontologies, the ontology introduced in [6] is more complete in terms of localisation and temporality. It also considers the required representational basis for environmental changes (e.g., events). However, what is missing is again an alignment to an upper level ontology, and although it has been claimed that the ontology is per se generic enough to be used in different domains, it is not obvious how it relates to commonly used upper ontologies and standards.

There are also few approaches in the literature proposing more general ontologies for IoT-related domains, regardless of their applications. The Semantic Sensor Network (SSN) ontology [21] is introduced as a generic and reusable knowledge model for sensor-related environments. The first version of SSN[1] was relying on the upper level ontology DUL[2] and took a remarkable step towards the reusability of ontologies [21]. However, the representational details in the first version of SSN could cause an excessive processing query time. That is why some lighter instantiations of SSN, such as IoT-Lite have been introduced [9]. Due to its light weight (in terms of expressivity), IoT-Lite allows us to define some relevant IoT-related concepts to support interoperability of heterogeneous sensor data. Although the vocabularies used in IoT-Lite are aligned with their generalized counterparts, the representation of the key concepts in sensor-related environment that are discussed in section 2.1 such as sensor, action and observation, is limited [26].

Regarding the load of the representational details in SSN bringing up excessive processing time, the W3C Spatial Data on the Web Working Group has proposed an updated version of SSN as a W3C recommendation with no import of the DUL ontology as its basis [12]. The new version of SSN provides a basis for some required concepts (e.g., sensor, observation, platform, etc.) in representing a smart environment. Although the new SSN is not based on DUL, a specific alignment module is also provided, which can be used if needed.

In this paper, we propose a generic ontology for smart (sensorized) environments (with at least one inhabitant/user) based on SSN. There are a number of works in the literature inspired by the old version of SSN [22, 23]. However, since the basis of our proposed ontology is the new version of SSN, we do not go through the details of old-SSN-inspired work, as their differences with our proposed ontology will be similar to the differences between the old and new version of SSN. In our design, we rely on

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

[2] DOLCE Ultra Light: www.ontologydesignpatterns.org/ont/dul/DUL.owl

the general concepts given in the new version of the SSN ontology and extend/specialize them in order to cover all the representational requirements given in section 2.1. This extension, which will be explained in the remaining sections of the paper, is focusing on both static (such as objects, deployment or network set-up) and dynamic (such as temporal and spatial, activity and event) concepts.

Regarding the design pattern approach, the ODP techniques underpinning our approach make it similar to the work proposed in [25], which introduces a generic Stimulus-Sensor-Observation ontology design pattern for representing observation-based data on the Semantic Web, although their focus is much more limited. Moreover, the focus of the work presented in [26] is on designing a core domain IoT ontology and proposes a reasonable categorization of high level concepts. What makes our approach different, however, is first its more comprehensive coverage of concepts, and also the provided representational details of the aforementioned aspects of a smart environment.

As said above, our proposed ontology can be seen as a network of modules whose basis are extracted from SSN. Each module is represented in the form of a pattern, as general as possible with the least possible dependencies on the other patterns. Doing so helps us to realize the main links in the eventual ontology. We can consequently make a stable and at the same time flexible network of concepts that can be updated with the minimum change propagation on the entire ontology, and where individual patterns can also be used in isolation for some specific reasoning tasks (e.g., in order to avoid issues with reasoning complexity or clashes in the relations to foundational ontologies).

3. Use Case: Ecare@Home Project

In order to set the stage and explain the background of our work, we will here briefly describe the project where the ontology modules were developed. We also introduce a reasoning example that will be used throughout the paper to exemplify the use of the modules.

The E-care@home project aims at providing a comprehensive IoT-based healthcare system, including state of the art communication protocols and high-level analysis of data from various types of sensors, combined with information from personal health records, and other background information, both generic and specific to a person. The main scenario is that of an elderly person who has some specific needs and potential medical conditions, but is still living at home. In order to increase the safety of the person, and reduce the frequency of appointments needed at a care facility, the patient and caregivers have agreed to fit the patients home with some sensors and a communication device, such as a tablet with a specific application installed. The challenge is to integrate and reason over all the information both from the sensors and the medical records at once, and derive the most likely conclusion, e.g., the current situation that the patient is in, the cause of some events, and the best action for the system to take next. This is quite a typical scenario for sensor-based monitoring systems, hence, it has enabled us to generalise our specific requirements (mentioned earlier in section 2.1) and provide a solution that we believe is highly reusable by other systems, regardless of the domain where such situation awareness and monitoring is required.

3.1. A Sample Scenario

Throughout this paper we will use the following example to illustrate our modules:

Let us assume we are monitoring a patient with severe COPD (Chronic Obstructive Pulmonary Disease). Since one effect of the disease is lung function reduction, patients tend to have a hard time to remain active. A reduced level of activity in their daily lives may be an indication of a worsening of the condition. The level of activity is furthermore seen as an important contextual background information for interpreting readings from medical sensors and self assessments, such as perceived breathlessness or oxygen saturation. Therefore one task of the E-care@home system is to create a time-line of the patient's daily activities. One possible (simplified) example of such an inference, based on sensor observations could be that the patient is watching TV as long as the person is seated on the couch in the living room and the TV is on. For performing this inference, we need at least two *sensors* attached to *objects* located in the *living room*: one that registers the couch occupancy and one that records the on/off status of the TV. Moreover, we need information about the patient, as an *actor* (or agent) whose activities can be observed through *sensing* processes implemented by these sensors. We also need several layers of abstraction in terms of observed events, i.e., both

low-level manifestations, such as that the TV is on, and complex events that are composed of (sequences or sets) of such manifestations, such as the notion of watching TV, as well as reasoning mechanisms to derive the latter from the former.

4. SmartEnv Ontology – Overview

In the previous section, we discussed the existing ontologies that have inspired our work, and the reasons why none of them cover all the requirements. In this section, we briefly provide an overview of the overall ontology network that we propose as a solution to this problem. Figure 1 represents an abstract overview of the overall ontology in terms of its modules and their relations. The figure is somewhat simplified in order to be more readable, e.g., most inverse relations have been omitted as well as some of the alignments to external classes and properties. In the following section we then go into details of the individual modules and their axiomatization.

The SmartEnv ontology[3] is composed of 8 ontology modules which are publicly available. Each module is representing a single principal feature of a smart environment. The requirements for the modules of the SmartEnv ontology are formalized based on the reasoning requirements, derived from the high-level requirements presented earlier in section 2.1. As we will see in the following subsections, most of the modules are extending other ontologies such as SSN and DUL. The name of the concepts and relations taken from the aforementioned ontologies are represented in the form of shortened IRI referring to the OWL definition of the concepts.

In this section we only provide the Description Logic (DL) notations of the classes and properties. To further clarify, in section 5.1 we provide examples showing how these DL notations are used to populate the SmartEnv ontology.

*4.1. **Time** Pattern*

The pattern **Time**[4] represents any temporal entities that we may use to represent things in a smart environment. In order to represent the temporal aspect of such environments, we have designed the pattern **Time** which is mainly an extension of the OWL-Time ontology, a W3C recommendation for describing temporal concepts [10].

The OWL-Time ontology provides precise representation for temporal entities in the form of either time instant or temporal duration. In the context of smart environments, we, however, require more specific temporal representation that allows us to also represent relative temporal distance (for example, between an event and its preconditions). By relative, we mean the temporal distance is indicated without knowing the specific timestamps located within a specific distance to a given time point (e.g., the event cooking is recognized at time-stamp `t` only if we know at least between 20 to 10 seconds before that, someone has been around the stove, regardless of the exact time `t`).

For this, we have extended the OWL-Time ontology and introduce it as our **Time** ontology pattern. In this pattern, we define three types of temporal entities representing time instants, time intervals and temporal distances. In the following, we go to the details of the classes, where the two prefixes `owl-time` and `se-time` refer to the URIs of the OWL-Time ontology, and our **Time** ontology pattern, respectively:

se-time:TemporalEntity is subsumed by the class `owl-time:TemporalEntity`:

$$\texttt{se-time:TemporalEntity} \sqsubseteq \texttt{owl-time:TemporalEntity} \tag{1}$$

It is also equivalent to the union of the three classes `se-time:Instant`, `se-time:Interval` and `se-time:TemporalDistance` as follows:

$$\texttt{se-time:TemporalEntity} \equiv \texttt{se-time:Instant} \sqcup \texttt{se-time:Interval} \sqcup \texttt{se-time:TemporalDistance} \tag{2}$$

se-time:Instant According to OWL-Time, an instance of the time instant is assumed to represent a temporal entity with zero extent or duration. The subsumption of this class in our **Time** pattern, i.e., `se-time:Instant`, is specialized to also include a date-time-stamp value supposed to be modeled by this class as follows, where the prefix `xsd` also refers to *http://www.w3.org/2001/XMLSchema*:

$$\texttt{se-time:Instant} \sqsubseteq \texttt{owl-time:Instant} \sqcap \texttt{se-time:TemporalEntity} \sqcap = 1\ \texttt{owl-time:inXSDDateTimeStamp.(xsd:dateTimeStamp)} \tag{3}$$

[3] https://w3id.org/smartenvironment/smartenv.owl

[4] https://w3id.org/smartenvironment/patterns/time.owl

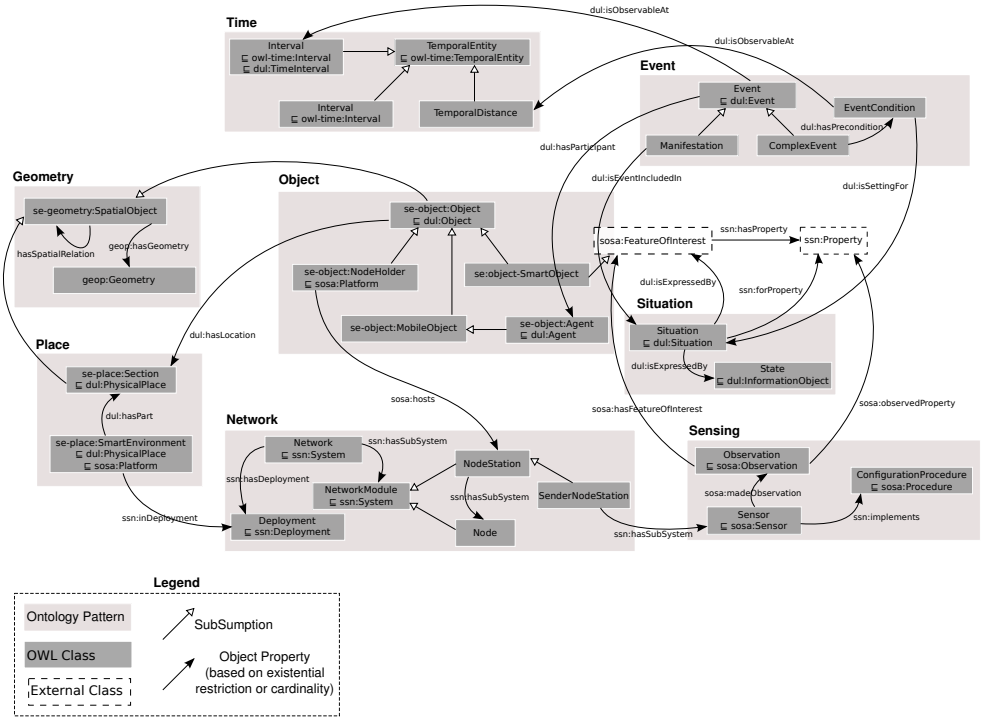

Fig. 1. SmartEnv Ontology based on its 8 building blocks in the form of ontology patterns.

se-time:Interval is likewise a specialization of the class `owl-time:Interval` and represents a temporal entity with an extent or duration. Our specialization includes more representational details of a duration whose starting time is known as follows:

$$\begin{aligned} &\texttt{se-time:Interval} \sqsubseteq \texttt{owl-time:Interval} \sqcap \\ &\quad \texttt{se-time:TemporalEntity} \sqcap \\ &\quad = 1\ \texttt{owl-time:hasBeginning.(se-time:Instant)} \sqcap \\ &\quad = 1\ \texttt{owl-time:hasDuration.(se-time:TimeDuration)} \end{aligned} \tag{4}$$

se-time:TimeDuration which is used in the aforementioned DL expression is a simple specialization of the class `owl-time:TimeDuration`. Given the fact that the class `owl-time:TimeDuration` represents the duration of a temporal extent expressed as a number scaled by a temporal unit, we can model the class `se-time:TimeDuration` to also be able to provide both the temporal length along with the temporal unit of the duration as follows:

$$\begin{aligned} &\texttt{se-time:TimeDuration} \sqsubseteq \texttt{owl-time:TimeDuration} \\ &\textit{where}: \\ &\texttt{owl-time:TimeDuration} \sqsubseteq \\ &\quad \texttt{owl-time:TemporalDuration} \sqcap \\ &\quad = 1\ \texttt{owl-time:numericDuration.(xsd:decimal)} \sqcap \\ &\quad = 1\ \texttt{owl-time:unitType.(owl-time:TemporalUnit)} \end{aligned} \tag{5}$$

se-time:TemporalDistance is finally the temporal concept that is needed to represent an interval whose starting-time in terms of a date-time value is unknown and is set relative to another time-position, with a specific distance. More specifically, a temporal distance is used when we need to explain a temporal constraint for an event or an activity whose preconditions need to be captured during a specific period of time located at certain distance with the time stamp of the event. In the **Time** pattern, we have defined this class in the form of two separate temporal durations with the same ending point as follows:

$$\begin{aligned} &\texttt{se-time:TemporalDistance} \sqsubseteq \\ &\quad \texttt{se-time:TemporalEntity} \sqcap \\ &\quad \exists\ \texttt{owl-time:before.se-time:Instant} \sqcap \\ &\quad = 2\ \texttt{owl-time:hasDuration.(se-time:Duration)} \end{aligned} \tag{6}$$

The object property `owl-time:before` indicates that the time instant is the target time that is temporally positioned not behind the two durations. Figure 2 represents how we define a temporal distance using

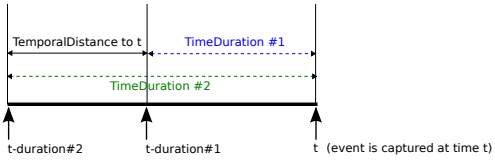

Fig. 2. Representation of relative temporal distance calculated based on 2 time durations from a given time point t.

two time durations temporally located before the given time-stamp `t` at which an event is expected to occur.

4.2. *Geometry* Pattern

Apart from the temporal aspect, in a sensorized environment, specifically when there are mobile agents such as robots, the representational model needs to also cover the spatial aspects of entities including the topology of objects, rooms, etc. For instance, there are many situations where we need to localize objects relative to the physical position of the user. For this, encoding the spatial relations between entities is essential (e.g., *"the kitchen is next to the living-room"* or *"the living room is located at the right side of the robot"*.). For this, we have designed a pattern called **Geometry**[5] relying on the upper level spatial-related ontology GeoSPARQL [27] and the Open Time and Space Core Vocabularies [10]. The OGC GeoSPARQL standard together with the Open Time and Space Core Vocabulary Specification (which provides qualitative directional relations) define an adequate vocabulary for representing geospatial data enabling qualitative spatial reasoning based on geometrical computations. The extension that we made upon these two ontologies is given in the following, where the prefixes `se-geometry`, `geop` and `ns`, refers to the **Geometry** pattern, GeoSPARQL ontology and the Open Time and Space Core Vocabularies, respectively:

se-geometry:SpatialObject is subsumed by the class `geop:Feature`. According to GeoSPARQL, a feature represents a spatial object that has a geometry. We constrain the definition by saying that each feature is also in a spatial and directional relations with at least one another spatial object:

$$\texttt{se-geometry:SpatialObject} \sqsubseteq \texttt{geop:Feature} \sqcap \exists\ \texttt{geop:hasGeometry.geop:Geometry} \sqcap \exists\ \texttt{se-geometry:hasSpatialRelation.se-geometry:SpatialObject} \quad (7)$$

se-geometry:hasSpatialRelation is an object property subsuming all the basic RCC-8 [28] topological relations (e.g., $\texttt{geop:sfContains} \sqsubseteq \texttt{se-geometry:hasSpatialRelation}$, $\texttt{geop:sfOverlaps} \sqsubseteq \texttt{se-geometry:hasSpatialRelation}$, etc.). Furthermore, this object property subsumes a more specialized object property called `se-geometry:hasDescriptiveSpatialRelation`, which is defined to subsume all the qualitative spatial relations (including descriptive directional relations) in the Open Time and Space Core Vocabularies (such as `ns:northOf`, `ns:nextTo`, `ns:near`, etc.):

$$\texttt{se-geometry:hasDescriptiveSpatialRelation} \sqsubseteq \texttt{se-geometry:hasSpatialRelation} \quad (8)$$

Where (due to the lack of space we have excluded the prefix `se-geometry` from all the following entities):

$$\top \sqsubseteq \forall\ \texttt{hasSpatialRelation}^{-}.\texttt{SpatialObject}$$
$$\top \sqsubseteq \forall\ \texttt{hasSpatialRelation.SpatialObject}$$

4.3. *Situation* Pattern

A "situation" illustrates a specific *state* of a "feature of interest" (e.g., the temperature of the living room is *warm*). By feature of interest we refer the concept defined in the SSN ontology as an object which is the interest of the observation process. Although states are usually time dependent, we decided to keep the representation of a situation as abstract as possible for the sake of generality. The concept of situation can be augmented with the concept of time in other patterns such as event-related patterns which are associated with temporal properties (see Section 4.8).

In order to represent various situations (related to feature of interests) in a smart environment we have designed the pattern **Situation**[6] that contains two classes State and Situation where the individuals of the former express the individuals of the latter class. In the following, we describe the representational details of these two classes, where the prefixes `se-situ`, `dul`, `ssn` and `sosa` refer to the URIs of the pattern **Situation**, DULCE, SSN and SOSA[7] ontologies, respectively:

[5] https://w3id.org/smartenvironment/patterns/geometry.owl

[6] https://w3id.org/smartenvironment/patterns/situation.owl

[7] SOSA is one of the ontologies imported to SSN [12]

se-situ:State represent a class whose individuals are assumed to declaratively express a feature of interest regardless of how this expression is realized. According to DUL, the class `dul:InformationObject` allows us to define any piece of information such as a text, a word, etc., independently from how it is concretely realized. We found this definition suitable to be specialized and be the basis of the class `se-situ:State` as follows:

$$\texttt{se-situ:State} \sqsubseteq \texttt{dul:InformationObject} \sqcap \exists\ \texttt{dul:expresses.se-situ:Situation} \quad (9)$$

se-situ:Situation is subsumed by the class `dul:Situation`. A situation in a smart environment can be more specialized and expressed by a specific property of a feature of interest (i.e., the class `sosa:FeatureOfInterest`, e.g., temperature of the living room) and its state (e.g., warm). Therefore, the definition of the class also includes three axioms that determine the relations between a feature of interest, its property and its relevant state. These axioms rely on the object property `dul:isExpressedBy`:

$$\begin{aligned}\texttt{se-situ:Situation} \sqsubseteq\ & \texttt{dul:Situation} \sqcap \\ & \exists\ \texttt{dul:isExpressedBy.sosa:FeatureOfInterest} \sqcap \\ & \exists\ \texttt{dul:isExpressedBy.se-situ:State} \sqcap \\ & \exists\ \texttt{ssn:forProperty.ssn:Property}\end{aligned} \quad (10)$$

4.4. *Sensing* Pattern

A sensing process is simply defined as the process of monitoring a specific property of a feature of interest using a sensing device. In order to represent such concept, we have designed the pattern **Sensing**[8] which is highly relying on the SSN ontology allowing us to model establishment of a sensing process. In the following we explain the classes as the specialization of concepts in SSN where the prefixes `se-sensing`, `sosa` and `ssn` refer to the URI of the pattern **Sensing**, SOSA (used in SSN) and the SSN ontologies, respectively:

se-sensing:Sensor is the specialization of the class `sosa:Sensor`. This class is designed to represent any sensing device used in an observation process with its own configuration that includes the sampling rate, data type of its outputs, etc.

$$\texttt{se-sensing:Sensor} \sqsubseteq \texttt{sosa:Sensor} \sqcap \exists\ \texttt{sosa:madeObservation.se-sensing:Observation} \sqcap \exists\ \texttt{ssn:implements.se-sensing:ConfigurationProcedure} \quad (11)$$

se-sensing:Observation is subsumed by the class `sosa:Observation` and represents a monitoring process that has been setup to observe the behavior of a feature of interest.

$$\texttt{se-sensing:Observation} \sqsubseteq \texttt{sosa:Observation} \sqcap \exists\ \texttt{sosa:madeBySensor.se-sensing:Sensor} \sqcap \quad (12)$$

It is worth mentioning that the information about the feature of interest, its properties, sensors' results, etc., is provided by the definition of the class `sosa:Observation` given in the SSN ontology [12].

se-sensing:ConfigurationProcedure as a procedure allows us to define specific configuration required for each sensor to be used in an observation process. These configurations can include setting up the sampling rate of the sensor, the types of sensor data, etc., that can be defined according to the application. A very generic definition of this class is given in the following:

$$\texttt{se-sensing:ConfigurationProcedure} \sqsubseteq \texttt{sosa:Procedure} \sqcap \exists\ \texttt{ssn:implementedBy.se-sensing:Sensor} \quad (13)$$

4.5. *Place* Pattern

The meaning of a place in the context of smart environments is twofold. First, by a place we mean the entire smart environment which holds the deployment of a sensor network and might also be composed of several sections. The second meaning of a place refers to each section of the main place with a specific identity that can be as such seen as a location for different objects. Given this preliminary definition, the pattern **Place**[9] defines a place as a specialization of the class `dul:PhysicalPlace` with the following details, where the prefixes `se-place`, `sosa`, `ssn` and `dul` refer to the URI of the pattern Place, SOSA (used in SSN), SSN and DUL ontologies, respectively:

se-place:SmartEnvironment represents the entire environment as a physical place (i.e., `dul:PhysicalPlace`) that is also assumed to hold a deployment of a sensor network. This class,

[8] https://w3id.org/smartenvironment/patterns/sensing.owl

[9] https://w3id.org/smartenvironment/patterns/place.owl

therefore, has to also be subsumed by the class `sosa:Platform`, which according to SSN, provides a link to a deployment process (i.e. an instance of the class `ssn:Deployment`):

```
se-place:SmartEnvironment ⊑                                    (14)
  dul:PhysicalPlace ⊓
  sosa:Platform ⊓
  ∃ dul:hasPart.se-place:Section
```

The information required to define an instance of the `se-place:SmartEnvironment` as a platform used in a deployment process is provided by the class `sosa:Platform` in the SSN ontology [12].

se-place:Section represents spatial sections as parts of a smart environment. Each section defines a physical place that can accommodate different objects. Furthermore, each spatial section in a smart environment has a geometry and therefore, can be in spatial relations with the other sections. In order to reflect such properties, we have specialized the definition by adding a subsumption relation with the class `se-geometry:SpatialObject` defined in the **Geometry** pattern:

```
se-place:Section ⊑  dul:PhysicalPlace ⊓                        (15)
  se-geometry:SpatialObject ⊓
  ∀ dul:isLocationOf.dul:PhysicalObject ⊓
  ∃ dul:isPartOf.se-place:SmartEnvironment
```

4.6. *Network* Pattern

A network in a smart environment is defined as a system containing different types of devices such as nodes and node stations. By node, we mean a communication module that indicates either a sending or a receiving data module in a network. It is worth mentioning that the current design of the Network Pattern only supports the request/response communication paradigm.

Each node depending on its type can be a part of a node station representing another type of device that contributes in establishing a network. Each node station contains a node along with other things including a sensor, power supplies, batteries etc.

The whole process of a network set-up regardless of its exact technical details is represented by a non-physical concept called deployment. The pattern **Network**[10] unifies the representation of environment automation installations that can be found in different systems. In the following we give the definition of the concepts in the **Network** pattern, where the prefixes `se-network` and `ssn` refer to the **Network** pattern and the SSN ontology, respectively:

se-network:Deployment extends the class `ssn:Deployment` and explains a platform (e.g., a smart environment) upon which a network is deployed:

```
se-network:Deployment ⊑  ssn:Deployment ⊓                      (16)
  ∃ ssn:deployedSystem.se-network:Network
```

The information related to the platform is inherited from the superclass `ssn:Deployment` in the SSN ontology [12].

se-network:Network is defined as a special type of system that has a deployment and is composed of a number of subsystems as follows:

```
se-network:Network ⊑  ssn:System ⊓                              (17)
  ∃ ssn:hasDeployment.se-network:Deployment ⊓
  ∃ ssn:hasSubsystem.se-network:NodeStation
```

se-network:NetworkModule describes the network modules as a system in the form of node stations and nodes contributing in sending and receiving data within a sensor network in the context of a smart environment:

```
se-network:NetworkModule ⊑  ssn:System                         (18)
```

se-network:NodeStation represents a network module categorized into two types of data sending module (i.e., `se-network:SenderNodeStation`) and data receiving module (i.e., `se-network:ReceiverNodeStation`). Each node station as a system (`ssn:System`) is located on a platform (e.g., as we will see in section 4.7 a node holder) in the environment and holds a number of nodes as its sub-systems:

```
se-network:NodeStation ⊑                                       (19)
      se-network:NetworkModule ⊓
      ∀ ssn:hasSubSystem.se-network:Node
```

se-network:Node likewise represents a network module that either in the form of `se-network:DataReceiverNode` or a

[10] https://w3id.org/smartenvironment/patterns/network.owl

se-network:DataSenderNode plays a role in transferring data:

$$\texttt{se-network:Node} \sqsubseteq \texttt{se-network:NetworkModule} \quad (20)$$

se-network:DataReceiverNode as its name indicates, models a node (as part of a node station) which receives data coming from sensors (or more specifically from sender modules):

$$\texttt{se-network:DataReceiverNode} \sqsubseteq \texttt{se-network:Node} \sqcap \exists\ \texttt{se-network:receivesDataFrom.se-network:DataSenderNode} \quad (21)$$

se-network:DataSenderNode also models a node which as a part of a node station sends sensor data (to a receiver):

$$\texttt{se-network:DataSenderNode} \sqsubseteq \texttt{se-network:Node} \sqcap \exists\ \texttt{se-network:sendsDataTo.se-network:DataReceiverNode} \quad (22)$$

se-network:ReceiverNodeStation defines a node station which holds a data receiver node as its part (sub-system):

$$\texttt{se-network:ReceiverNodeStation} \sqsubseteq \texttt{se-network:NodeStation} \sqcap \forall\ \texttt{ssn:hasSubSystem.se-network:DataReceiverNode} \quad (23)$$

se-network:SenderNodeStation likewise represents a node station which is assumed to contain both a data sender node and also sensing devices (sensors):

$$\texttt{se-network:SenderNodeStation} \sqsubseteq \texttt{se-network:NodeStation} \sqcap \forall\ \texttt{ssn:hasSubSystem.se-network:DataSenderNode} \sqcap \exists\ \texttt{ssn:hasSubSystem.sosa:Sensor} \quad (24)$$

se-network:receivesDataFrom is an object property providing the relation between a se-network:DataReceiverNode and a se-network:DataSenderNode. It is also the inverse of the object property se-network:sendsDataTo as follows (the prefix se-network has been excluded):

$$\top \sqsubseteq \forall\ \texttt{receivesDataFrom.DataSenderNode} \quad (25)$$
$$\top \sqsubseteq \forall\ \texttt{receivesDataFrom}^{-}\texttt{.DataReceiverNode}$$
$$\texttt{receivesDataFrom} \equiv \texttt{sendsDataTo}^{-}$$

*4.7. **Object** Pattern*

The pattern **Object**[11] allows us to define objects based on their important features or abilities in the context of smart environments. The class dul:PhysicalObject provides a suitable representational basis for the objects' taxonomy, which we have categorized into two types of smart objects and node holders. By smart object we refer to those objects that are the interest of an observation process (i.e, feature of interest). Due to the usual difficulties of installing sensors in a smart home, it is common to use some other objects (i.e. node holders) to host sensors. This separation provided by this pattern is specifically useful for other computational modules such as a configuration planner one of whose tasks is checking the status/functionality of sensors by sending a robot to their locations.

Each smart object (or a feature of interest) has at least a property to be observed. Another categorization of smart objects that has been considered in the object pattern, is about their mobility. An objects is considered as mobile only if its location as one of its properties, can change. In order to also be able to reflect the spatial relations between objects (e.g., the "fridge is connected to the cupboard"), or between an object and a place (e.g., "the bed is located at the left side of the bedroom"), it is required to define objects in a smart environment also as a se-geometry:SpatialObject defined in the pattern **Geometry** (see Section 4.2).

In the following, we represent the DL notations of each object type along with their properties, where the prefixes se-object, dul and ssn refer to the **Object** pattern, DUL and SSN ontologies, respectively:

se-object:Object as the main class of the **Object** pattern is subsumed by the class dul:PhysicalObject. The class definition is specialized according to the aforementioned description about different types of objects that we can have in a smart environment:

$$\texttt{se-object:Object} \sqsubseteq \texttt{dul:PhysicalObject} \sqcap \texttt{se-geometry:SpatialObject} \sqcap \exists\ \texttt{dul:hasLocation.dul:PhysicalPlace} \quad (26)$$

[11] https://w3id.org/smartenvironment/patterns/object.owl

se-object:SmartObject as an object also considered as a feature of interest due to its property/properties which is/are the interest of an observation process. For this to be represented, we have defined a smart object also as a subclass of `sosa:FeatureOfInterest` which provides the relation between the object and its properties:

$$\texttt{se-object:SmartObject} \sqsubseteq \texttt{se-object:Object} \sqcap \texttt{sosa:FeatureOfInterest} \quad (27)$$

se-object:NodeHolder also as an object is also considered as a platform that can host (hold) a device (a node) in a sensor network. In this way, we can differentiate between the objects, namely those that are the interest of the observation, and the objects which, as the holder of sensors, are used for sensor localization processes.

$$\texttt{se-object:NodeHolder} \sqsubseteq \texttt{se-object:Object} \sqcap \texttt{sosa:Platform} \quad (28)$$

se-object:MobileObject defines a movable object that can be found at different places in the environment. In other words, there is an observable property of interest called "Location" for mobile objects:

$$\texttt{se-object:MobileObject} \sqsubseteq \texttt{se-object:Object} \sqcap \exists\ \texttt{ssn:hasProperty.se-object:Location} \quad (29)$$

where[12]

$$\texttt{se-object:Location} \sqsubseteq \texttt{sosa:ObservableProperty} \sqcap \exists\ \texttt{ssn:isPropertyOf.se-object:MobileObject}$$

se-object:Agent The class `se-object:MobileObject` can be further specialized and form another type of objects that are able to be proactive and participate in some events. The class `se-object:Agent` subsumed by the class `dul:Agent` allows us to represent inhabitants of an environment (e.g., humans, pets, etc.), that can be involved in an activity or an event (e.g., a person is an agent at home as he/she is often involved in various activities such as "sitting on couch"). Each agent can own, as its constituent, some objects (i.e., `se-obeject:Object`) whose location depends on the location of the agent. More specifically, a constituent can express other objects physically attached to the agent. However, a constituent does not need to be permanently attached to the agent. For instance, a chair might be deemed as a constituent as long as it is held by the agent (this relation to the constituent is borrowed from `se-dul:Agent`).

$$\texttt{se-object:Agent} \sqsubseteq \texttt{dul:Agent} \sqcap \texttt{se-object:MobileObject} \sqcap \forall\ \texttt{dul:isParticipantIn.dul:Event} \quad (30)$$

4.8. *Event* Pattern

The pattern **Event**[13] is an extension of the representation of events in DUL. In this extension we have defined two different types of events including a manifestation and complex event. By manifestation, we refer to those events that can be directly captured from sensor data and represent the occurrence of a smart home situation through a sensing process. However, the latter event type, as its name indicates, represents more complicated events whose occurrence depends on several preconditions [23]. Each precondition as such represents a specific situation assumed to be observed within an interval with a specific temporal distance to the event's occurrence time. Furthermore, the pattern **Time** which is per se based on the OWL-Time ontology, can provide the required basis to represent the temporal properties of the smart environment to capture changes in the form of events or activities. In the following we provide the representation of each class where the prefixes `se-event`, `se-time` and `dul` refer to the patterns **Event**, **Time** and the DUL ontology:

se-event:Event as a general event class is subsumed by the class `dul:Event`. According to DUL, each event is expected to be observable at/within a `dul:TimeInterval`. On the other hand, the pattern **Time** provides a general representation for temporal entities including time interval and temporal distances. Furthermore, as mentioned in Section 4.7, an event can have an agent (a proactive object) as its participant. Given the aforementioned explanations, we define the class `se-event:Event` as follows:

$$\texttt{se-event:Event} \sqsubseteq \texttt{dul:Event} \sqcap \forall\ \texttt{dul:hasParticipant.dul:Agent} \sqcap \quad (31)$$

[12] According to the SSN ontology: $\texttt{sosa:ObservableProperty} \sqsubseteq \texttt{ssn:Property}$

[13] https://w3id.org/smartenvironment/patterns/event.owl

$$\exists \text{ dul:isObservableAt.se-time:Interval}$$

In order to complete the above class definition, we have also defined the class `se-time:Interval` from the pattern **Time** as the sub class of the `dul:TimeInterval`:

$$\text{se-time:Interval} \sqsubseteq \text{dul:TimeInterval} \quad (32)$$

se-event:Manifestation as a specialized version of the class `se-event:Event` indicates a situation (as we will see later, it is more specifically a `se-situ:Situation`) directly captured from sensor data:

$$\text{se-event:Manifestation} \sqsubseteq \text{se-event:Event} \sqcap \exists \text{ dul:isEventIncludedIn.dul:Situation} \quad (33)$$

se-event:ComplexEvent as a specialized version of the class `se-event:Event` and represents an event whose occurrence depends on capturing a number of preconditions represented as situations:

$$\text{se-event:ComplexEvent} \sqsubseteq \text{se-event:Event} \sqcap \exists \text{ dul:hasPrecondition.se-event:EventCondition} \quad (34)$$

se-event:EventCondition as a `dul:Situation` represents preconditions of a complex event (in the form of a situation: `se-situ:Situation`) which are needed to be captured within a specific temporal distance from the time-stamp of the complex event, where by temporal distance we refer to (`se-time:TemporalDistance`) (see section 4.1):

$$\text{se-event:EventCondition} \sqsubseteq \text{dul:Situation} \sqcap \exists \text{ dul:isPreconditionOf.se-event:ComplexEvent} \sqcap \exists \text{ dul:isObservableAt.se-time:TemporalDistance} \sqcap \exists \text{ dul:isSettingFor.dul:Situation} \quad (35)$$

5. SmartEnv Ontology – Construction

In this section, we show how by integrating the 8 separate ontology module, the SmartEnv ontology[14], representing different aspects of a smart environment, is constructed (see Figure 1).

The SmartEnv ontology is formed as the result of importing all the 8 modules. The connection between each pair of modules is accomplished by specializing the definition of concepts in each module and then linking them together. For instance, the class `se-network:Deployment` used in the pattern **Network** is further specialized via the link to the class `se-place:SmartEnvironment` in the pattern **Place**:

$$\text{se-network:Deployment} \sqsubseteq \exists \text{ ssn:deployedOnPlatform.se-place:SmartEnvironment}$$

All the specialized relations between modules illustrated in Figure 1, are also listed in Table 1. For further clarification, in the following we exemplify the population process of SmartEnv for representation of a smart home as an example of a smart environment.

5.1. SmartEnv Population

We have developed SmartEnv in E-care@home to be used as the representation model of the data that is gathered from a deployment test bed in Örebro University called Ängen, a sensorized apartment which provides functional facilities for the research development including Ambient Intelligence (AMI) solutions. One of the relevant applications of context reasoning in E-care@home is Activity recognition of Daily Living (ADL) as well as monitoring of other features of interest such as the physiological and health-related parameters of the users. For this, we have equipped Ängen with a number of both environmental[15] (e.g., pressure sensor, light sensor, motion sensor, etc.) and medical sensors. Before running the observation process, we need to initialize the ontology with all the equipments including the network modules, objects, their locations as well as the activities or events that we are interested to recognize. In our preliminary set-up we equipped Ängen with 14 different sensors at different sections (rooms) of Ängen. The following axioms are examples of statements used in SmartEnv population. For the sake of declarativity, these examples are either in the form of a unary predicate (class instantiation) or a binary predicate (positive object property assertions):

It is worth mentioning that apart from the A-box, the population also includes a process of extension of the T-box in order to define more specialized subclasses of SmartEnv. We first define Ängen in the ontology as a smart environment which as a living place of an elderly

[14] https://w3id.org/smartenvironment/smartenv.owl

[15] Due to the ethical concerns, we have avoided using cameras in E-care@home.

Table 1
Linked modules in SmartEnv ontology.

Module(1) Name	Module(2) Name	Axiom
Network	Place	se-network:Deployment ⊑ ∃ ssn:deployedOnPlatform.se-place:SmartEnvironment
Network	Object	se-network:NodeStation ⊑ ∃ sosa:isHostedBy.se-object:NodeHolder
Event	Object	se-event:Event ⊑ ∃ dul:hasParticipant.se-object:Agent
Event	Situation	se-event:Manifestation ⊑ ∃ dul:isEventIncludedIn.se-situ:Situation
Event	Situation	se-event:EventCondition ⊑ ∃ dul:isSettingFor.se-situ:Situation
Object	Event	se-object:Agent ⊑ ∃ dul:isParticipantIn.se-event:Event
Object	Place	se-object:Object ⊑ ∃ dul:hasLocation.se-place:Section
Object	Network	se-object:NodeHolder ⊑ ∃ sosa:hosts.se-network:NodeStation
Place	Network	se-place:SmartEnvironment ⊑ ∃ ssn:inDeployment.se-network:Deployment
Place	Object	se-place:Section ⊑ ∃ dul:isLocationOf.se-object:Object
Network	Sensing	se-network:SenderNodeStation ⊑ ∃ ssn:hasSubSystem.se-sensing:Sensor

person is composed of one living room, 2 bedrooms, one bathroom and one kitchen.

⋆ Smart Environment Description:

```
se-place:SmartEnvironment(ängen).
LivingRoom ⊑ se-place:Section
BedRoom ⊑ se-place:Section
BathRoom ⊑ se-place:Section
Kitchen ⊑ se-place:Section
LivingRoom(livingroom1).
Kitchen(kitchen1).
BathRoom(bathroom1).
BedRoom(bedroom1).
BedRoom(bedroom2).
dul:hasPart(ängen, livingroom1).
dul:hasPart(ängen, kitchen1).
dul:hasPart(ängen, bathroom1).
dul:hasPart(ängen, bedroom1).
dul:hasPart(ängen, bedroom2).
```

Given the structure of the environment, we continue the population process with the objects, their properties and locations. Due to the lack of space, we focus only on a single object (a couch) and show how we represent it's location with a pressure sensor attached to it to monitor the couch occupancy:

⋆ Object Description:

```
Couch ⊑ se-object:SmartObject
Couch(couch1).
ssn:Property(pressure).
ssn:hasProperty(couch1, pressure).
dul:hasLocation(couch1, livingroom1).
```

Next, we define the network such as its deployment in Ängen, nodes and node stations:

⋆ Network Description:

```
se-network:Network(network1).
se-network:Deployment(deployment1).
ssn:inDeployment(ängen, deployment1).
se-network:NetworkModule(nodeStation1).
ssn:hasSubSystem(network1, nodeStation1).
```

The network is deployed in order to implement the observation process which is initialized by assigning the sensors to the feature of interests, or in other words, smart objects with their properties, which in our case is the pressure on the couch:

⋆ Sensing Description:

```
PressureSensor ⊑ se-sensing:Sensor
PressureSensor(sensor1).
ssn:hasSubSystem(nodeStation1, sensor1).
se-sensing:Observation(observation1).
sosa:hasFeatureOfInterest(observation1, couch1).
sosa:observedProperty(observation1, pressure).
```

Activities or events as the realization of situations in the environment need to be defined in the ontology. For this, we first define the two possible situations related to the couch and its pressure property:

⋆ Situation Description:

```
se-situ:Situation(couchPressed).
dul:isExpressedBy(couchPressed, couch1).
dul:isExpressedBy(couchPressed, on).
se-situ:Situation(couchUnPressed).
dul:isExpressedBy(couchUnPressed, couch1).
dul:isExpressedBy(couchUnPressed, off).
```

Then observation processes associated with an inference process is able to report the timestamps at which an event or whatever that can change the situation of the environment, occur. For this to be possible, we have to define activities (e.g., sitting on the couch) in the ontology based on the given situations. Assuming the sitting activity is realized when the pressure sensor attached to the couch is triggered, the ontology is populated as follows:

⋆ Event Description:

```
Sitting ⊑ se-event:ComplexEvent
SittingCondition ⊑ se-event:EventCondition
dul:hasPrecondition(Sitting, SittingCondition).
dul:isSettingFor(SittingCondition, couchPressed).
se-time:TemporalDistance(distance).
se-time:TimeDuration(duration1).
```

```
se-time:TimeDuration(duration2).
owl-time:numericDuration(duration1, 0).
owl-time:numericDuration(duration2, 0).
owl-time:hasDuration(distance, duration1).
owl-time:hasDuration(distance, duration2).
dul:isObservableAt(SittingCondition, distance).
```

The mentioned above axioms define a temporal distance (i.e., `distance`) which is defined based on two time durations (`duration1` and `duration2` both with length zero (i.e., they are referring to the same time point t (see Figure 2)). In other words, the precondition of the event sitting needs to be captured exactly when the sitting activity occurs. Once the pressure sensor is triggered we immediately (with no delay) infer the sitting event.

Given the populated ontology as explained above we can start observing the environment where each sensor is assigned to a feature of interest to detect changes and consequently recognize the related events/activities. In order to map the stream of sensor data into our representation model, we need a reasoner. Depending on the features of the domain, the level of uncertainty and complexity, we may apply different reasoning method (monotonic or non-monotonic). Due to the inherent uncertainty of sensor systems, we decided to apply a non-monotonic reasoner which is based on stable model semantics [13]. The details how the ontological axioms are converted to the rules understandable by the non-monotonic reasoner (AnswerSet Solver) which is out of the scope of this paper, can be found in [14].

The mentioned above population process can be extended in order to cover all the objects, sensors and the equipments existing in the environment. During one of our test runs, the SmartEnv ontology was first populated with the static information about the Ängen set-up which totally resulted in 172 specialisations (i.e. subclasses) of the ODP classes, and 203 individuals. Given this instantiated ontology, the observation process is then able to feed the ontology with the instances of manifestations corresponding to the changes detected in sensor outputs. For instance, monitoring a person doing different sorts of activities (such as cooking, watching TV, etc.) in Ängen resulted about 200 additional individuals, describing the situations captured from the environment. These individuals, which are related to different classes including the manifestation and the subclasses of the complex event class, makes the ontology reasoning-ready for different purposes, such as configuration planning or context recognition in multi-inhabitant environments. The example scenario outlined in Section 3 is only one among a multitude of activities that are relevant to detect in the context of the E-care@home project.

6. Discussion & Future Work

In this paper, we proposed a network of ontology patterns targeting the representational aspects (such as sensing process, network configuration, objects' taxonomy, event representation, topological representation, etc.) of smart environments. The final ontology is formed by integrating the proposed patterns representing a smart environment. In order to avoid the complex design issues as the result of dependencies between the representational aspects, we have applied the ODP approach as an incremental methodology in designing ontologies. The ODP approach allows us to start by first creating general and small patterns for each aspect and then link them together. In this way, regardless of its size, the ontology becomes flexible for further updates with the least possible change cost.

The majority of the ontology modules constituting the SmartEnv ontology are mainly relying on SSN and DUL ontologies, however with a number of specializations, either in the form of extension of class hierarchies or updating links between concepts.

However, reusing existing vocabularies from SSN or DUL was not a straightforward process. There are a number of generic concepts whose definitions make them a suitable basis for other context-related concepts. For instance, the class `dul:InformationObject`) which is used as the super class of the class `se-situ:State`. Finding such generic concepts can be time consuming. A means to support the use of existing vocabularies (including both concepts and object properties) and/or patterns without the need for creating or importing such classes would ease the process to a considerable extent.

The network of ontology patterns may in the future be equipped with more patterns required to cover other aspects of a smart environment. One aspect that we are currently investigating concerns the habits of the users (inhabitants) whose definitions (in terms of preconditions) are not necessarily clear at the time when we populate the ontology. For this, as a next step, we may either generalize the event pattern or add a new ontology module to capture such concepts that allow us to relate some activities of the user to his/her habits based on their repetitions.

Acknowledgments:

The work is supported by the project E-care@home funded by the Swedish Knowledge Foundation 2015-2019.

References

[1] Nguyen, T. V. and Lim, W. and Nguyen, H. and Choi, D. and Lee, C.: Context ontology implementation for Smart Home. CoRR abs/1007.1273, (2010)

[2] Garvita Bajaj and Rachit Agarwal and Pushpendra Singh and Nikolaos Georgantas and Valérie Issarny: A study of existing Ontologies in the IoT-domain. CoRR, (2017)

[3] Ye, J., Stevenson, G. T., and Dobson, S. A.: A top-level ontology for smart environments. Pervasive and Mobile Computing, 7(3), 359-378, (2011). DOI: 10.1016/j.pmcj.2011.02.002

[4] Holder, L.B., and Wemlinger, Z.: The COSE ontology: bringing the semantic Web to smart environments. (2011).

[5] Chen, H., Finin, T. and Joshi, A.: An ontology for context-aware pervasive computing environments. Special Issue on Ontologies for Distributed Systems, Knowledge Engineering Review, Vol. 18, pp.197âĂŞ207 (2003).

[6] Abdulrazak, B. , Chikhaoui, B. , Gouin-Vallerand, C. , Fraikin, B.: A standard ontology for smart spaces. International Journal of Web and Grid Services. 6, 3, 244-268, (2010)

[7] Chen, H., Finin, T., Joshi, A.: The SOUPA ontology for pervasive computing. Ontologies for agents: Theory and experiences. 233–258, (2004)

[8] Bonino, D. and Corno, F.: Dogont - ontology modelling for intelligent domotic environments. Proceedings of the International Semantic Web Conference, Vol. 5318 of Lecture Notes in Computer Science, Springer, pp.790-803 (2008).

[9] Bermudez-Edo, M., Elsaleh, T., Barnaghi, P., Taylor, K. IoT-Lite: A lightweight semantic model for the Internet of things and its use with dynamic semantics. Personal Ubiquitous Comput. 21, 3 (June 2017), 475-487. DOI: https://doi.org/10.1007/s00779-017-1010-8

[10] Cox, S., Little, C. Time ontology in OWL. W3C recommendation, W3C, October 19 2017, URL: https://www.w3.org/TR/owl-time/ (2017)

[11] Sanchez, A., Jim Jones, J. Open time and space core vocabulary specification: https:http://www.observedchange.com/tisc/ns/ (2013)

[12] Haller, A., Janowicz, K., Cox, S., Phuoc, D.L., Taylor, K., and Lefrançois, M. Semantic sensor network ontology. W3C recommendation, W3C, October 19 2017, URL: https://www.w3.org/TR/vocab-ssn/ (2017)

[13] Gebser M., Kaminski R., Kaufmann B., Ostrowski M., Schaub T., Thiele S: Engineering an incremental ASP solver. In: Garcia de la Banda M., Pontelli E. (eds) Logic Programming. Lecture Notes in Computer Science, vol 5366. Springer, Berlin, Heidelberg (2008)

[14] Alirezaie, M. and Renoux, J. and Köckemann, U. and Kristoffersson, A. and Karlsson, L. and Blomqvist, E. and Tsiftes, N. and Voigt, T. and Loutfi, A.: An ontology-based context-aware system for smart homes: E-care@home. Sensors, vol 17. (2017)

[15] Gangemi, A. and Presutti, V.: Ontology design patterns. Handbook on ontologies. International Handbooks on Information Systems (2009)

[16] Blomqvist, E. and Hitzler, P. and Janowicz, K. and Krisnadhi, A. and Narock, T. and Solanki, M.: Considerations regarding ontology design patterns. Journal Semantic Web, 7, 1-7 (2016)

[17] Alexander, C. and Ishikawa, S. and Silverstein, M.: A pattern language: Towns, Buildings, Construction. (1977)

[18] Roger D. Eastman; Craig I. Schlenoff; Stephen B. Balakirsky; Tsai Hong Hong: A sensor ontology literature review.(NISTIR) - 7908 (2013)

[19] Tao, G. and Xiao, H.W. and Hung, K.P. and Da, Q.Z.: An ontology-based context model in intelligent environments. Int. Conf. on Communication Network and Distributed Systems Modeling and Simulation - 270–275 (2004)

[20] Nguyen, T.A. and Raspitzu, A. and Aiello, M.: Ontology-based office activity recognition with applications for energy savings. Journal of Ambient Intelligence and Humanized Computing - 667–681 (2014)

[21] Compton, M. and Barnaghi, P. and Bermudez, L. and Garcia-Castro, R. and Corcho, Ó. and Cox, S. and Graybeal, J. and Hauswirth, M. and Henson, C. and Herzog, A. and Huang, V. and Janowicz, K. and Kelsey, W. D. and Phuoc, D. L. and Lefort, L. and Leggieri, M. and Neuhaus, H. and Nikolov, A. and Page, K. and Passant, A. and Sheth, A. and Taylor, K.: The SSN ontology of the W3C semantic sensor network incubator group. Web Semantics: Science, Services and Agents on the World Wide Web. Elsevier, 17, (2012).

[22] Sezer, O. B. and Can, S. Z. and Dogdu, E.: Development of a smart home ontology and the implementation of a semantic sensor network simulator: (ICCTS), (2015)

[23] Alirezaie, M. and Loutfi, A.: Reasoning for improved sensor data interpretation in a smart home. ARCOE-Logic Workshop Notes, pp. 1-12, (2014)

[24] Cox, Simon J.D.: Ontology for observations and sampling features, with alignments to existing models. Semantic Web Journal pp. 1-18, (2016)

[25] Janowicz, K. and Compton, M.: The stimulus-sensor-observation ontology design pattern and its integration into the semantic sensor network ontology. Workshop on Semantic Sensor Networks, SSN (2010)

[26] Seydoux, N. and Drira, K. and Hernandez, N. and Monteil, T.: IoT-O, a core-domain IoT ontology to represent connected devices networks: Int. Conf. on Knowledge Engineering and Knowledge Management (EKAW), 561–576, (2016)

[27] Battle, R. and Kolas, D.: Enabling the geospatial semantic web with Parliament and GeoSPARQL. Semantic Web Journal pp. 355–370, (2012)

[28] Cohn, A. G., Bennett, B., Gooday, J., Gotts, M.: Qualitative spatial representation and reasoning with the region connection calculus. GeoInformatica, 1, 275-316, (1997)