The RealEstateCore Ontology

Karl Hammar[1][0000−0001−8767−4136]✉, Erik Oskar Wallin[2], Per Karlberg[2], and David Hälleberg[3]

[1] Department of Computer Science and Informatics
Jönköping University, Sweden
karl.hammar@ju.se
[2] Idun Real Estate Solutions AB, Sweden
{erik,per}@idunrealestate.com
[3] Akademiska Hus AB, Sweden
david.halleberg@akademiskahus.se

Abstract. Recent developments in data analysis and machine learning support novel data-driven operations optimizations in the real estate industry, enabling new services, improved well-being for tenants, and reduced environmental footprints. The real estate industry is, however, fragmented in terms of systems and data formats. This paper introduces RealEstateCore (REC), an OWL 2 ontology which enables data integration for smart buildings. REC is developed by a consortium including some of the largest real estate companies in northern Europe. It is available under the permissive MIT license, is developed and hosted at GitHub, and is seeing adoption among both its creator companies and other product and service companies in the Nordic real estate market. We present and discuss the ontology's development drivers and process, its structure, deployments within several companies, and the organization and plan for maintaining and evolving REC in the future.

Keywords: ontology · smart buildings · building automation · IoT · energy optimization · space analytics

Resource Type: Ontology
IRI: https://w3id.org/rec/full/3.0/
DOI: https://doi.org/10.5281/zenodo.2628367

1 Introduction

Real estate companies today face new data integration demands, driven by both changing customer expectations in an increasingly digital market, and by societal challenges relating to sustainability and resource efficiency. Concretely, customers expect landlords to be able to communicate and interact digitally regarding the leased property and the equipment, furnishings, and systems within it. Increasingly, customers are requesting access to building systems and data streams in order to themselves carry out different types of analytics and optimization. Simultaneously, real estate companies operate on competitive markets and thus need to reduce operating expenditures, e.g., facility management costs, inventory management costs, and energy costs. Reducing energy utilization is a particularly important goal since it is both one of the most substantial costs, and one which immediately affects the organisation's environmental footprint.

The use of sensor and actuation systems together with data integration and analytics platforms (augmented by recent advantages in machine learning) show great promise in enabling companies to meet these demands. A building equipped with such systems and platforms is sometimes referred to as a *Smart Building*; future visions of city planners and real estate developers often revolve around *Smart Cities* made up of such Smart Buildings. In this vision, buildings will for instance regulate heating and cooling systems based not only on the number of inhabitants at the present; but also on future numbers anticipated based on historic trends and signals from the surrounding city (e.g., from the public transport system); and on anticipated temperatures and wind speeds from weather forecasts. The building systems will be able to inform the landlord when maintenance is due or when a subsystem breaks; it will inform the tenant when they are using their leased spaces inefficiently, and allow them to integrate physical and digital access control; it will enable external service providers access to parts of the physical and digital building, as required to provide their services.

All of these Smart Building features require that data from and about the building can be exchanged in standardized formats using agreed-upon meaning. Semantic Web ontologies are an obvious technology solution to enable such exchange. There have been several attempts to develop such ontologies for the Smart Buildings domain. However, these ontologies have typically been developed based on the needs of distinct but narrow groups: ontology engineers, building engineers, hardware or systems developers; or for comparatively narrow use cases such as assisted living scenarios. In this paper we present the RealEstateCore[4] (REC) ontology: the first, to our knowledge, ontology developed specifically by real estate companies for their needs. RealEstateCore is developed according to agile Ontology Engineering best practices; it is modular, to avoid ontological over-commitment and enable customization; it is free for anyone to use, licensed under the permissive MIT license; and *it works*, being used in commercial products and being deployed to represent 100.000+ data signals in 12 buildings covering over 220.000 m^2.

The paper is organized as follows: Section 2 describes prior work in this field, and positions our contributions against said work; Section 3 describes RealEstateCore, how it was developed, and some distinguishing features; Section 4 presents the consortium behind RealEstateCore, and some companies and systems that use RealEstateCore today; and finally, Section 5 concludes the paper by discussing planned future developments for the ontology and associated API:s and tools.

[4] `https://realestatecore.io/`

2 Related Work

The RealEstateCore ontology spans across and draws influences from prior work in three distinct but related domains: digital representations of buildings and their constituent elements; control and operation of the building and its systems; and emerging IoT ontologies.

Data in the first domain is typically represented using the *Industry Foundation Classes* (IFC) data model, an established and open ISO standard developed in the late 1990s by the buildingSMART industry group. IFC supports interoperability among a variety of BIM and CAD tools [3]. The standard has been evaluated and found to aid in data exchange in building design and planning processes [2,8]. IFC has been translated into and made available as an OWL representation, *ifcOWL*, as described in [4]. IFC/ifcOWL is a very large and comprehensive standard, covering 1300+ classes and 1500+ properties, including a large amount of content that from a sensor data integration perspective is superfluous. The building structure needs that we have revolve primarily around navigating the building parthood hierarchy, which is only a small subset of the functionality provided in ifcOWL. Consequently, the RealEstateCore consortium has elected to not reuse ifcOWL as is, but rather to align REC to IFC concepts as needed, to allow imports of existing IFC models.

There are three major legacy standards in building automation that a Smart Building system would need to be able to interoperate with; *KNX* (commonly used in Europe), *BACnet* (commonly used in the US), and *Modbus* (commonly used in industry factory settings). All three are available in different dialects and versions, some of which can operate over IP networks, others of which use other communications protocols but can be bridged to IP networks using specialized hardware. The Brick Schema [1] is an OWL ontology that models the categories of equipment and measurable properties that might be found in such systems, based on the notion of *tagsets* (roughly comparable to an OWL class and in Brick represented as such) and *points* (roughly comparable to a sensor). The model is elegant, generic, and proven; the RealEstateCore consortium is working on establishing REC-Brick alignments, for increased interoperability between the two models. However, Brick only covers metadata about a building system installation: to this, REC adds features that are required to interact with said installation, e.g., device configuration, observation and actuation messages, etc.

The *Semantic Sensor Networks* (SSN) [10] and *Sensor, Observation, Sample, and Actuator* (SOSA) [12] ontologies are well-established and integrated models covering the sensor/actuator domains. SOSA covers the fundamentals of observations, sensors, features of interests, actuations, processes, results, etc., and SSN extends on this with concepts relating systems of devices, deployments of systems, system capabilities, input and output to procedures, etc. The original SSN model (sans SOSA) was developed for ontology engineers. In developing REC we considered adopting that original SSN model as a foundation, but our intended users (software developers in real estate companies) found the model to be too complex for them to apply; we thus developed a REC-specific device model. However, the more recent editions of SSN cited above, which extend SOSA, have been significantly simplified, and we are looking to integrate REC with SSN/SOSA going forward. SSN/SOSA complements REC with more expressive semantics for processes, sampling, and features; REC complements SSN/SOSA with domain knowledge relevant to the real estate sector, e.g., the types of devices, communications protocols, and measurements used.

The *Smart Appliances REFerence* (SAREF) ontology [6] is another contender in this space. As the name indicates, SAREF provides a model covering Smart Appliances, e.g., home appliances for residential use. RealEstateCore targets a broader class of systems, typically installed in the building itself, and used for building-wide analytics and optimization.

The reader will note that RealEstateCore does not claim to improve on the modeling state-of-the-art in any of these domains; instead, our value is in providing a comprehensive Semantic Web ontology model that covers and integrates all of them, based on actual real estate owners' needs.

3 The RealEstateCore Ontology

3.1 Initial Use Cases

RealEstateCore was initially developed in support of two specific use cases; energy usage analysis and optimization; and presence analysis.

Energy Usage Analysis and Optimization Energy costs are a substantial part of the operational costs of running a building. In order to reduce these costs, one needs to reduce both total energy utilization over time and the momentary peak power load. Typically, utilities companies charge by both of these measures, and the tariffs are set by the latter. A reduced operational cost causes an increase in operational net income, which in standard accounting models is immediately linked to the book-keeping value of the property in question; i.e., reducing the energy utilization of a building *immediately* increases its value. Additionally, of course, reducing energy use also reduces environmental impact.

There are numerous examples of how machine learning systems can be used to operate heating/cooling systems in buildings, with energy gains of 40 % being reported [7]. However, doing this type of analysis and optimization on a large scale, spanning the full portfolio of a real estate owner, is very difficult. The common case is that real estate owners have several different building automation systems, making the installation of an energy reducing system into a substantial and different project for each real estate and its building automation system[5].

Normalizing data (e.g., using the REC ontology) from different building automation systems makes it possible to apply an energy reducing system on a large

[5] At Vasakronan, a key REC sponsor and user, a September 2018 inventory of the deployed building automation systems identified more than 10 different archetypes (climate control, access control, fire alarm, elevator control, etc.) and up to 30 different vendors and version combinations — it's a mess.

scale. In order to reduce energy consumption different data-driven methods can then be applied, such as:

- Finding broken or misaligned equipment (sensors reporting outlier values).
- Adapt the heating/cooling/ventilation/lightning of the rooms in the building after the actual need (i.e., by detecting momentary occupancy and load).
- Anticipating future needs and running systems based accordingly. E.g., *What is the prediction of needed heating/cooling tomorrow/in half a day/in the next hour?* Systems might stop the cooling already by noon on Friday, if historically the office has cleared out by 3 o'clock.
- Utilizing thermo-dynamical effects. Can the characteristics of the building's material be used to store heat or cold? E.g. to use bare concrete walls or ceilings to accumulate energy (when price is lower) to be discharged to achieve the intended climate.

Presence Analysis As indicated above, a key factor in operating a building system efficiently is the ability to detect to what degree different parts of the building are occupied by people. Important metrics include the number of people present in parts of the building, people flows through the building (i.e., from where and to where are they moving), what activity is being carried out (high- or low-intensity), etc. Presence analysis is also important in optimizing the design of building spaces for better usage, based on people densities, flows, etc. Understanding and adapting to the behavior of tenants and their customers in this manner has started to become vital to the business offering, and for the real estate owner's long term planning of how to manage their portfolio.

Presence can be detected or deduced not only from dedicated infrared sensors or cameras, but also from other types of more commonly deployed sensors (air quality sensors, sound sensors, etc.) and even from infrastructure that has traditionally not been used for such purposes (Wifi usage, coffee machine energy use via electricity meters, bathroom water flows using water meters, etc.). This enables presence analytics in a building using an already installed building automation system's sensors, hence reducing the need to install new dedicated presence sensors, saving costs and time.

3.2 Development Process and Priorities

RealEstateCore has been under development since 2016, by a team of academics and real estate owners in collaboration (see Section 4). The development process has been shaped by the need to rapidly reach a state where the ontology can be deployed and used by software developers with limited ontology engineering experience, for the purposes discussed above. A minimum viable product perspective has thus been employed, and when we have needed to balance between different quality aspects, clarity and usability have been given priority over expressiveness, reusability, or metaphysical grounding. Implications of this prioritization include our class and property naming strategies (employing cognitively relevant common-sense terms, avoiding premature generalization), the use of single-domain and single-range properties (the semantics of multiple `rdfs:range` or `rdfs:range` declarations can be unclear to this category of users [11, pp. 127–128]) and our choice not to build on established foundational ontologies (which can be difficult for non-expert users to understand and maintain [11, pp. 177–181]). To streamline communications with this class of users we have found WIDOCO [9] and WebVOWL [14,16] to be very helpful tools.

The minimim viable product approach marries well to the eXtreme Design [5] (XD) ontology engineering method, which we have employed in an adapted format. Our initial attempts at executing a "pure" XD process, based on composing the ontology from small reusable Ontology Design Patterns (ODPs), was unsuccessful; there simply weren't enough ODPs of sufficient quality, relevant to the domain we were modeling, to support our development process. We considered generalizing our own designs and creating a pattern library out of them, but the overhead costs associated with this process were deemed prohibitive. Instead, we chose to reuse known good solutions as built-in components of our ontology implementation, and to adopt other aspects of the XD method:

Requirements management XD's recommendation to formalize requirements by way of user stories, competency questions (CQ:s), contextual statements, and reasoning requirements, have helped structure and delimit the modeling problems. Some example CQ:s are given in Table 1; the full set is provided as annotations on the individual ontology modules[6].

Modular development While ODPs haven't been used, the domain we cover has been split apart into larger modules, enabling flexibility in ontology composition, or the development of custom modules, by a deploying organization.

Pair programming We have had a minimum of two developers jointly work on the modules under development. These developers have been selected such that they represent both the ontology engineering disciplines and the domain disciplines covered by the modeling challenge at hand.

Release early, release often Every iteration has been passed to the software developer teams for testing and deployment; on several occasions those developers have pushed back and required changes to simplify system development. We have applied semantic versioning principles such that removal or renaming of ontology content (i.e., breaking changes) have resulted in the ontology and its modules being assigned new major version numbers [13,15]; as a consequence of this the ontology is, at the time of writing, already at version number 3.0, even though all initial use-cases have not yet been fully implemented in the ontology (see Section 5).

In accordance with the XD method, the requirements that were prioritized by the customers were developed first. Consequently, several features relating to the presence analysis use case remain in our backlog at the time of writing.

[6] We have used the `cpannotationschema:coversRequirements` annotation property for this purpose; while it was originally designed to cover CQ requirements on ODPs, we have found no more suitable vocabulary for expressing CQ:s over ontologies.

Table 1. RealEstateCore Competency Questions excerpt.

Question	**Module**
Which buildings and land make up the real estate Ulvsnaes 1:7?	Core
Where is the real estate Ulvsnaes 1:7 located?	Core
Which parts of Building 1 (e.g., rooms, wings, etc) are covered by electricity meter 3?	Core
What equipment is mounted in server rooms in Building 7?	Building
Are there any labs in Building 7?	Building
Which facade- or roof-mounted sensors are installed in Building 7?	Building
What kind of device is AHU731?	Device
How is device AHU731 connected to the building infrastructure (comms bus, protocol, protocol version, connection parameters, etc)?	Device
What addressable components (sensors, actuators, etc) make up device AHU731?	Device
Which sensor did observation OBS-846294-PID22-88 come from?	Device
What are all the values reported over the past 24 hours from temperature sensors in rooms 2, 4, and 7?	Device

We align REC to existing semantic resources whenever feasible, using established alignment predicates from OWL, RDFS and SKOS[7]. For non-semantic resources we create namespaces and mint IRIs based on identifier values from those resources (e.g., `https://w3id.org/rec/alignments/Haystack/sensor`). The alignments are versioned and published in the project GitHub repository[8].

The systems that are used in the real estate business typically have a long lifetime, on the order of 20+ years. It is important that the REC ontology is robust enough to continue working for similar time periods. As we cannot risk link rot over time breaking the ontology, we have chosen to avoid direct dependencies on external resources (i.e., `owl:imports` against non-REC ontologies). Instead, we redefine those external classes and properties that we reuse (GeoSPARQL, Dublin Core, VANN, etc.) within our own ontology modules. While this may be unorthodox it is not formally illegal, and we believe it to be necessary to ensure the longevity and maintainability of the ontology.

3.3 Ontology Description

RealEstateCore is constructed as two base modules (*Metadata* and *Core*), and several domain-specific modules extending this base via `owl:imports` predicates:

- *Metadata:* Includes annotation properties that are used to document the ontology (from Dublin Core[9], CreativeCommons[10], and VANN[11]).

[7] `owl:equivalentClass`, `owl:equivalentProperty`, `rdfs:seeAlso`, `owl:sameAs`, `skos:related`, etc.

[8] `https://github.com/RealEstateCore/rec`

[9] `http://purl.org/dc/elements/1.1/`

[10] `http://creativecommons.org/ns#`

[11] `http://purl.org/vocab/vann/`

- *Core:* Collects the top-level classes and properties that span over or are reused within multiple REC modules. Imports the Metadata module, and is in turn imported by all other specific child modules.
- *Agents:* Covers basic types of agents (people, organizations, groups), structurally aligned with FOAF[12].
- *Building:* Covers types of building components and rooms.
- *Device:* Covers different device types (sensors and actuators), device configurations, device actuation, etc.
- *Lease:* Covers lease contracts, types of leasable premises, etc.

The RealEstateCore ontology imports all of these modules. In total, the ontology contains 141 classes, 56 object properties, 63 data properties, and 181 individuals. Visualizing and communicating the design of an ontology of this size is a challenge; Figure 1 provides a partial schema diagram covering some key classes and Figure 2 lists the top-level classes and properties. For a detailed specification we refer the reader to the documentation[13]. The ontology and its documentation are both made available via content negotiation at the ontology IRI `https://w3id.org/rec/full/3.0/`. Additionally, in accordance with community practice, the ontology has been indexed in LOV[14].

Many of the design choices when implementing REC were straight-forward, based on our understanding of the domain and problem space. However, several were not; in particular, the notions of `QuantityKind`, `PlacementContext`, and `DeviceFunctionType`.

QuantityKind Our use cases deal largely with sensor observations, which are typically grouped by *measurement category* (e.g., *"temperature"*) and quantified into some *measurement units* (e.g., C°). These two attributes are disjoint; there isn't a 1–1 mapping between measurement categories and measurement units, and in fact many measurement units are reused across different measurement categories. Thus both need to be represented and related to sensor observations.

To model this, we considered creating a subclass hierarchy of different types of observations based on the measurement category to which each observation belonged, e.g., `‘‘:obs1 rdf:type :TemperatureObservation; :hasUnit :celsiusUnit . :TemperatureObservation rdfs:subClassOf :Observation’’`. However, this design would constrain our ability to use subclass relations for other types of as-yet unforeseen categorization, and it would also require T-box modifications to support new types of observations, which reduces modifiability and reusability. As the name indicates, we instead found a solution in the `QuantityKind` concept from the QUDT ontology[15], which is defined as follows: *"A Quantity Kind is any observable property that can be measured and quantified numerically. Familiar examples include physical properties such as length, mass,*

[12] `http://xmlns.com/foaf/spec/`
[13] `https://doc.realestatecore.io/3.0/full/`
[14] `https://lov.linkeddata.es`
[15] `http://qudt.org/schema/qudt/`

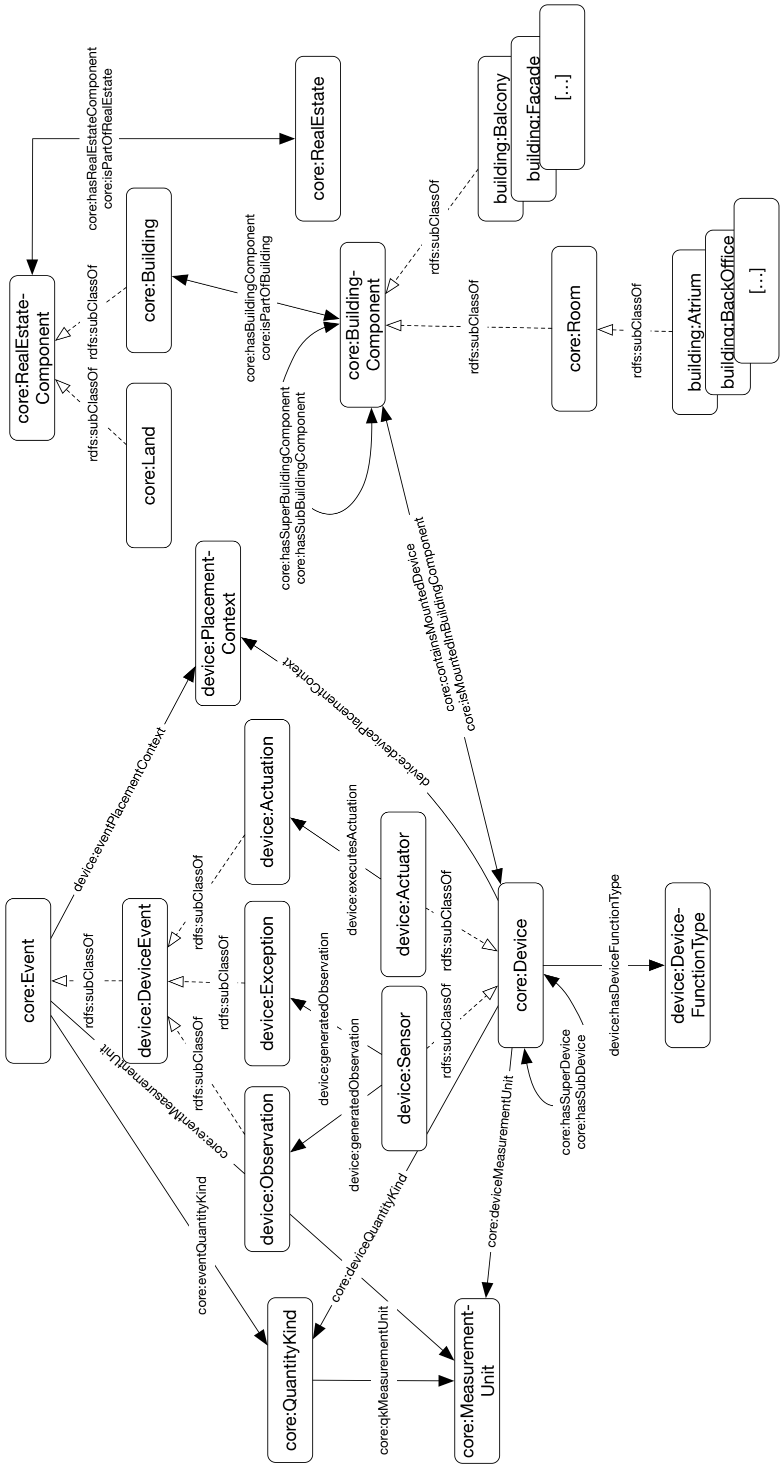
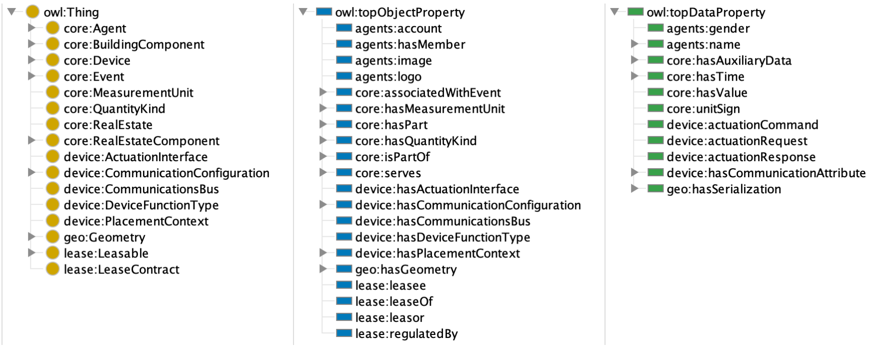


Fig. 1. Partial schema diagram of the RealEstateCore ontology.



- owl:Thing
  - core:Agent
  - core:BuildingComponent
  - core:Device
  - core:Event
  - core:MeasurementUnit
  - core:QuantityKind
  - core:RealEstate
  - core:RealEstateComponent
  - device:ActuationInterface
  - device:CommunicationConfiguration
  - device:CommunicationsBus
  - device:DeviceFunctionType
  - device:PlacementContext
  - geo:Geometry
  - lease:Leasable
  - lease:LeaseContract

(a) Classes

- owl:topObjectProperty
  - agents:account
  - agents:hasMember
  - agents:image
  - agents:logo
  - core:associatedWithEvent
  - core:hasMeasurementUnit
  - core:hasPart
  - core:hasQuantityKind
  - core:isPartOf
  - core:serves
  - device:hasActuationInterface
  - device:hasCommunicationConfiguration
  - device:hasCommunicationsBus
  - device:hasDeviceFunctionType
  - device:hasPlacementContext
  - geo:hasGeometry
  - lease:leasee
  - lease:leaseOf
  - lease:leasor
  - lease:regulatedBy

(b) Object Properties

- owl:topDataProperty
  - agents:gender
  - agents:name
  - core:hasAuxiliaryData
  - core:hasTime
  - core:hasValue
  - core:unitSign
  - device:actuationCommand
  - device:actuationRequest
  - device:actuationResponse
  - device:hasCommunicationAttribute
  - geo:hasSerialization

(c) Data Properties

Fig. 2. RealEstateCore top-level constructs.

time, force, energy, power, electric charge, etc.[...]". We standardize on this notion and in the ontology provide a number of named `QuantityKind` instances relevant to our domain, e.g., `Temperature`, `ActiveEnergyL1`, `CurrentL1`, `CO2`, etc. QuantityKind is the "what" that is being measured or acted upon.

PlacementContext Sensor and actuator positions in a building are typically described either in terms of spatial placement, or in terms of meronymic placement. Often, both types of placement information are used concurrently. However, in understanding how a sensor or actuator can be analysed or used, a different type of placement information is typically needed, one that describes the *functional placement* of the device (i.e., what part of a process that this sensor or actuator has been placed to measure and/or affect). We attempted to model this type of information using established methods, but found not suitable vocabularies covering our needs. Accordingly, we developed the class `PlacementContext` to cover this need, and provide a set of named instances for it, e.g., `PrimaryCoolingFlow`, `ExhaustAir`, `ElectricalGridIntake`, etc. PlacementContext is the "where" something is being measured or acted upon. The use of QuantityKind and PlacementContext in combination provides a large flexibility to describe different sensing and actuation situations.

DeviceFunctionType We need to be able to represent a large number of device types in buildings, e.g., boilers, dampers, air diffusers, elevators, escalators, etc. Each of these may contain nested devices: different types of sensors, actuators, control units, etc., which in turn might contain even further nested devices. We thus have diverging granularities for the members of the `Device` subclass hierarchy. Trying to reconcile both type hierarchies under one joint root, we found that this made for a very confusing ontology for our intended users – with concepts in close proximity in the hierarchy that in the real world most considered to be quite far apart. We thus opted to model the top-level device typing via an object property `hasDeviceFunctionType`, a corresponding class `DeviceFunctionType`, and a set of named individuals: `Compressor`, `Fan`, `Pump`, `Elevator`, etc.

4 RealEstateCore Usage

The REC ontology has been under development since mid-2016. In late 2017 the RealEstateCore consortium was founded by the majority developers of the ontology; three real estate companies (Vasakronan, Akademiska Hus, and Willhem), one software development company (Klipsk), and two academic partners (RISE and Jönköping University). Initially the consortium operated under a non-binding membership charter; in early 2019, it initiated processes to become an own legal entity, enabling it to apply for R&D grants from funding agencies.

Participation in the consortium is open to anyone interested in contributing to the project's goals. There is a membership fee for corporations (for 2019 set at 1000 €) but individual personal memberships are free-of-charge. There is no need to be a member in order to use or contribute to the RealEstateCore ontologies – all source is freely available on GitHub and is licensed under MIT license – but in order to hold offices within the project (e.g., on the release engineering team, marketing and communications team, etc.) a membership is required.

In the follow section we describe some known uses of RealEstateCore.

4.1 Vasakronan: The Idun Platform

Vasakronan is the leading property company in Sweden, with a focus on commercial properties in major growth regions. Vasakronan owns and manages 174 properties with a total area of approximately 2.4 million m^2 – the portfolio is valued at ca 13.3 billion €. Vasakronan prioritizes environment and climate work; it is a carbon neutral company and is ISO 14001 certified.

The *Idun* smart building platform started as an internal Vasakronan development project, before being spun out into its own commercial startup. Over the past three years, more than 20 man-years of development have been invested into the platform. Idun converts data inputs from devices in the buildings into REC-compliant messages using edge-based servers, and integrates this data to support consumption by external (telemetry streams, time series analytics, etc.) and internal (state models, actuation facilities, etc.) consumers. The platform is built on the open sourced Microsoft Azure IoT Edge framework, which can run either on physical computers in the buildings or at some cloud service provider. When provisioned as a SaaS offering, Idun by default uses a mixture of dedicated physical servers, a private cloud, and the Azure cloud.

Three interfaces are provided for users/systems to consume data from Idun:

- A streaming API for firehose access to the sensor data streams.
- A REST API that provides telemetry data and enables knowledge graph manipulation and actuation.

- An analysis and reporting interface; provides access to a RealEstateCore-based analysis and reporting environment using Microsoft Power BI.

The telemetry messages that are delivered by edge nodes all adhere to the REC ontology (i.e., `Observation`, `Actuation` and `Exception`), but are comparatively terse. When these messages are passed into the system, they carry an authenticated device identity, provided by the edge node. This device identity is used to look up additional device metadata (position, type, units, etc.) from the building knowledge graph, expressed per REC; this metadata is used to enrich the message for later analysis purposes. For a schematic overview of the Idun platform, see Figure 3.

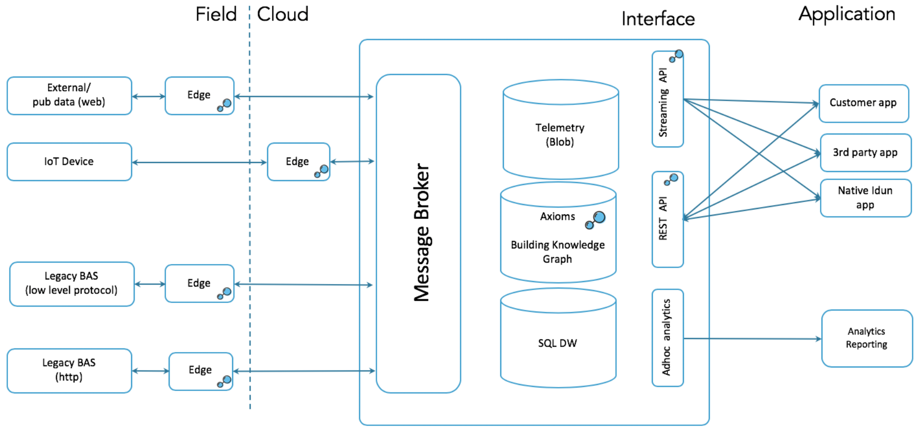


Fig. 3. Idun architecture – the blue overlay icon indicates REC-using components.

At the time of writing (April 2019) the Idun deployment within Vasakronan covers more than 80.000 physical sensors and actuators (each of which can generate a number of logical signals for consumption by the system) in more than 10 buildings totaling circa 200.000 m^2. In addition another 20.000 signals from external energy reporting systems, weather forecast services, etc. are connected to Idun. Sampling rates depending on the performance capabilities of the underlying systems and necessity – ranging from milliseconds on electricity meters to 15 minutes for battery powered IoT sensors and once per 24 hour for external energy reporting systems. For traditional building automation systems the sample interval is typically once per minute. Idun puts a lot of development effort into streamlining the onboarding of existing buildings; within the next 12 months we expect to have connected 90% of Vasakronan's portfolio.

Presently there are some 15 developed modules that translate building automation system messages into REC semantics for consumption by Idun; and several more are being developed by actors in the REC ecosystem.

4.2 Akademiska Hus

State-owned Akademiska Hus (AH) owns and manages over 3.3 million m^2 of university and college facilities; AH has a market share of 60% of Swedish higher education institutions, with campuses including Sweden's oldest and most prestigious universities (total holdings valued at ca 8.2 billion €). The company has set a goal being carbon neutral in the building operation by 2025. To this end, AH is deploying REC for data integration of and analytics over real-time data streams covering energy performance and indoor climate.

The pilot deployment of REC is at a 13,434 m^2 office building on the KTH campus in Stockholm. The building was constructed in 2005, and it has several modern systems; hi-tech sensors and energy meters throughout, connection to a campus micro-grid for district heating, cooling, and electricity, dedicated systems for energy management, building management, and demand-control ventilation.

The energy management system measures energy utilization for heating-, cooling- and electricity, on subsystem or building level – with 145 sensors exporting data every 60 minutes. The Building Management System (BMS) measures and controls the installation system within the building. It controls signals, temperatures, pressures, and airflow using both sensors and actuators. Together with the Demand Control Ventilation (DCV) system, energy is distributed within the building to reach a set indoor climate or process. The BMS systems uses a PLC standard and collects 726 signals from sensors and actuators every second minute. The DCV system automatically adjusts supply air-flow to the actual demand (set-points) in the room/spaces that it acclimatizes. It monitors and controls 7,944 signals every minute on a 3,000 m^2 area (typically lecture halls and office- and conference rooms). These signals are all mapped to REC semantics using Azure IoT Edge, before being passed to analytics and infrastructure platforms that can consume REC (e.g., Idun).

In addition to this initial pilot deployment, AH has recently also deployed REC at a second site; the Natural Sciences Building at Umeå University, which generates 3,902 signals (from the DCV system) over 7,800 m^2.

4.3 Willhem

Willhem owns approximately 26,000 rental apartments (1.8 million m^2, worth ca 3.4 billion €) in 13 cities throughout Sweden, with headquarters in Gothenburg. Like Vasakronan and Akademiska Hus, reducing energy utilization is an important driver for Willhem adopting RealEstateCore; the company's long-term goal is to cut their energy use by 50 % compared to a 2011 baseline. This will require automating and optimizing energy systems in the buildings, which necessitates data integration. Willhem are at an early stage in their RealEstateCore deployment; they are presently prototyping solutions for how to translate data from an existing sensor infrastructure to REC notation, and how to subsequently consume and analyze that REC-coded data. A candidate technology component in this future workflow is the IoT platform ThingsBoard[16].

[16] https://thingsboard.io/

4.4 The Building Knowledge Project

The 460'000 € research project *Building Knowledge*, running 2019–2020, develops and evaluates methods for integrating semantic and machine learning technologies, for applications in the real estate sector. The project is closely aligned with the RealEstateCore initiative (five of the six REC founder organisations participate in the project), and it uses the ontology as a test-bed for tooling development, as well as contributes to REC API development.

4.5 Additional Usages

At present more than 10 different partners and suppliers are developing REC-based services or integrations; a subset are presented below.

- *Schneider Electric EcoStruxure integration:* Schneider Electric is a world leader in power management products. They have developed a REC connector for the product EcoStruxure Building Operation[17].
- *Sweco Elements:* Sweco is a leading European architecture and engineering consultancy. The product Sweco Elements is a 3D modeling engine for CAD/BIM planning work that supports REC data consumption for visualization. The internal data sources that drives the Elements product can also expose BIM data in a REC format for ecosystem participants to consume.
- *Flowity:* Flowity is a subsidiary of ÅF, an engineering and design company within the fields of energy, industry and infrastructure, with business and clients all over the world. Their AI-based camera platform analyses the flow of patronage and their behavior, an essential task in real estate based business intelligence. Flowity uses REC to model locations.
- *Metry:* Metry collects and structures consumption data from utility providers, smart meters and offline meters. The company focuses on collection, quality and structure, and manages data points for over a third of Sweden's largest real estate companies, such as Vasakronan, Catena, Rikshem and Kungsleden. Metry has started to implement REC in their data structure and APIs.

5 Future Work

RealEstateCore provides classes and properties that support modeling of devices, their configurations, capabilities, and the values that they report or messages that they receive; modeling of buildings, including components, room types, and locations; and modeling of (rudimentary) contractual situations relating to these buildings. These features enable data integration that supports other systems that provide dashboards and time-series analytics on building, floor, room, or tenant level, which in turn supports the energy optimization use cases discussed

[17] https://web.archive.org/web/20190405122638/https://github.com/BuildingsLabs/EboIoTEdgeConnector

in Section 3. However, we do not yet have classes or properties that cover higher-level concepts (e.g., present energy utilization state, energy-using hardware or processes, target values, prognoses, etc.). A key step in the near future is to increase the expressivity of the ontology to support such higher-level semantics.

As mentioned in Section 3.2, the same situation is true for the presence detection and analysis use-case; while we have the semantics in place to support analysis in other systems, we do not yet have first-order representations of detected people, flows, etc. in the RealEstateCore ontology itself. Without such features only device-facing API:s fully utilize the data integration potential afforded by ontology use; when these features are developed (likely using SSN/SOSA as discussed in Section 2), higher-order data integration tasks between compliant analytics platforms will be made possible. This is a highly prioritized strategic development for the consortium, as it will enable increased competition among analytics systems suppliers, and prevent vendor lock-in.

We have also identified a number of domains and use cases for which REC modules will need to be adapted or entirely new modules be constructed:

- *Leases and rentals:* The Lease module is rudimentary and needs work to support integration with established facilities and rental management systems.
- *Access control:* Supporting integrated authentication and authorization across both physical (buildings, rooms) and digital (IT systems) assets.
- *Inventory management:* Modeling non-Internet-connected equipment fitted in buildings, including configuration, operations, vendors, protocols, manuals, etc. This could cover everything from PA systems to swing sets.

Finally, while the use of REC as a shared vocabulary has enabled significant data integration gains already, the RealEstateCore consortium has come to realize that standardizing the API:s by which REC-compliant data is exchanged, is an equally important aspect. The Idun platform provides building graph[18] and streaming data[19] API:s for REC 2.3 that have become de-facto standards for REC deployment. These API:s, while fully functional, were not designed to accommodate typical semantic data characteristics, e.g., using IRI identifiers, or using data schemas (ontologies) that can differ from implementer to implementer depending on which modules are loaded. A highly prioritized development over the coming quarters (Q2-Q3 2019) is the development of a standard REC API; we are tentatively looking at GraphQL as potential foundation for that API.

References

1. Balaji, B., Bhattacharya, A., Fierro, G., Gao, J., Gluck, J., Hong, D., Johansen, A., Koh, J., Ploennigs, J., Agarwal, Y., et al.: Brick: Towards a unified metadata schema for buildings. In: Proceedings of the 3rd ACM International Conference on Systems for Energy-Efficient Built Environments. pp. 41–50. ACM (2016)

[18] `https://w3id.org/rec/api/2.3/graph/`

[19] `https://w3id.org/rec/api/2.3/streaming/`

2. Bazjanac, V., Crawley, D.: Industry foundation classes and interoperable commercial software in support of design of energy-efficient buildings. In: Proceedings of Building Simulation'99. vol. 2, pp. 661–667 (1999)
3. Bazjanac, V., Crawley, D.B.: The implementation of industry foundation classes in simulation tools for the building industry. Tech. rep., Lawrence Berkeley National Laboratory (1997)
4. Beetz, J., Van Leeuwen, J., De Vries, B.: Ifcowl: A case of transforming express schemas into ontologies. Ai Edam **23**(1), 89–101 (2009)
5. Blomqvist, E., Hammar, K., Presutti, V.: Engineering Ontologies with Patterns – The eXtreme Design Methodology. In: Hitzler, P., Gangemi, A., Janowicz, K., Krisnadhi, A., Presutti, V. (eds.) Ontology Engineering with Ontology Design Patterns: Foundations and Applications, pp. 23–50. IOS Press (2016)
6. Daniele, L., den Hartog, F., Roes, J.: Created in Close Interaction with the Industry: The Smart Appliances REFerence (SAREF) Ontology. In: Cuel, R., Young, R. (eds.) Formal Ontologies Meet Industry. pp. 100–112. Springer (2015)
7. Evans, R., Gao, J.: DeepMind AI reduces energy used for cooling Google data centers by 40%. `https://web.archive.org/web/20190322212318/https://blog.google/outreach-initiatives/environment/deepmind-ai-reduces-energy-used-for/` (July 2016), accessed: 2019-03-26
8. Froese, T., Fischer, M., Grobler, F., Ritzenthaler, J., Yu, K., Sutherland, S., Staub, S., Akinci, B., Akbas, R., Koo, B., et al.: Industry foundation classes for project management-a trial implementation. Electronic Journal of Information Technology in Construction **4**, 17–36 (1999)
9. Garijo, D.: WIDOCO: a wizard for documenting ontologies. In: International Semantic Web Conference. pp. 94–102. Springer (2017)
10. Haller, A., Janowicz, K., Cox, S.J., Lefrançois, M., Taylor, K., Le Phuoc, D., Lieberman, J., García-Castro, R., Atkinson, R., Stadler, C.: The modular ssn ontology: A joint w3c and ogc standard specifying the semantics of sensors, observations, sampling, and actuation. Semantic Web **10**(1), 9–32 (2019)
11. Hammar, K.: Content Ontology Design Patterns: Qualities, Methods, and Tools, vol. 1879. Linköping University Electronic Press (2017)
12. Janowicz, K., Haller, A., Cox, S.J., Le Phuoc, D., Lefrançois, M.: Sosa: A lightweight ontology for sensors, observations, samples, and actuators. Journal of Web Semantics **In Press** (2018)
13. Klein, M.C., Fensel, D.: Ontology versioning on the semantic web. In: SWWS. pp. 75–91 (2001)
14. Lohmann, S., Negru, S., Haag, F., Ertl, T.: Visualizing ontologies with VOWL. Semantic Web **7**(4), 399–419 (2016)
15. Preston-Werner, T.: Semantic versioning 2.0.0. `https://web.archive.org/web/20190321081743/https://semver.org/spec/v2.0.0.html`, accessed: 2019-03-26
16. Wiens, V., Lohmann, S., Auer, S.: WebVOWL Editor: Device-Independent Visual Ontology Modeling. In: Proceedings of the ISWC 2018 Posters & Demonstrations, Industry and Blue Sky Ideas Tracks. CEUR Workshop Proceedings, vol. 2180. CEUR-WS.org (2018)