Planned ETSI SAREF Extensions based on the W3C&OGC SOSA/SSN-compatible SEAS Ontology Patterns

Maxime Lefrançois
Univ. Lyon, Mines Saint-Étienne, CNRS, Laboratoire Hubert Curien UMR 5516,
F-42023 Saint-Étienne, France
maxime.lefrancois@emse.fr

ABSTRACT

Mid-June 2017, the ETSI SmartM2M working group voted two work items, DTS/SmartM2M-103548 and DTS/SmartM2M-103549, with the goal to enhance and augment the SAREF ontology with some of the design, development, and publication choices that have been made in the context of the ITEA2 SEAS (Smart Energy Aware Systems) project. This paper provides an overview of these choices and their rationale. In particular, we describe contributions regarding: (i) the design of the ontology as a set of simple core ontology patterns, that can then be instantiated for multiple engineering-related verticals; (ii) the design and publication of the SEAS modular and versioned ontology in conformance with the publication and metadata best practices, with the additional constraint that every term is defined under a single namespace. These planned additions to SAREF will ease its adoption and extension by industrial stakeholder, while ensuring easy maintenance of its quality, coherence, and modularity. Finally, because the SEAS ontology generalizes the future W3C&OGC SOSA/SSN (Sensor, Observation, Sensing, Actuation / Semantic Sensor Network) ontology, these work items contribute to the convergence of the different reference ontologies relevant for the IoT domain.

1 INTRODUCTION

We observe more and more interest from industrial stakeholders into the Semantic Web formalisms and technologies, in particular as they are foreseen as a means to reach semantic interoperability between heterogeneous systems. On the other hand, it is often hard to choose among the 600+ ontologies referenced on the Linked Open Vocabulary (LOV) the ones that are worth reusing. In fact, there is a high heterogeneity among these ontologies in terms of adoption, institutional statuses, structure/content/publication/metadata quality, maintenance, extensibility, or even availability. We claim that ontologies that aim at becoming a reference in the industry and the IoT should meet the highest quality criterias possible, while having an internal structure that is consistent and simple enough to be easily mastered by the application developers, who are not experts in Semantic Web technologies.

The ITEA2 Smart Energy-Aware Systems (SEAS) project[1] aimed at designing and developing a global ecosystem of services and smart things collectively capable of ensuring the stability and the energy efficiency of the future energy grid. The choice has been made to adopt Semantic Web formalisms to achieve semantic interoperability between the services and the smart things, and one of the tasks in the project was thus dedicated to the design of an ontology. On top of the fact that the ontology had to cover a large and heterogeneous set of domains, strict quality requirements stemmed from the industrial context of its development and use.

This paper provides an overview of the design, development, and publication choices that have been made in the development of the SEAS ontologies to meet all those requirements. These choices are driven by the following three major goals:

Design goal 1. *The SEAS ontology must conform as much as possible to the publication and metadata best practices that have been defined by the Semantic Web community. Violation of these best practices must be detected as soon as possible, and as little work as possible must be left to the ontology maintainers.*

Design goal 2. *The SEAS ontology must conform to quality criteria one are used to in open source development projects: it must be modular, use semantic versioning, its source must be open so that anyone can be able to comment, raise issues, or contribute.*

Design goal 3. *The SEAS ontology users must have a learning curve as short as possible, even though the width and depth of the domains to be covered is very big. In particular, it must be easy to map it to some object-oriented programming language. One corollary to this requirementis that every term must be defined under the same namespace.*

The rest of this paper is organized as follows. Section 2 first precises the resolution taken in terms of development and publication choices for the ontology with respect to these three goals. Section 3 gives an overview of the content of SEAS and its modular organization. Section 4 to 7 sequentially introduce the four engineering-centric core SEAS ontology patterns, and provides an overview of how they are instantiated to model knowledge from different vertical domains.

SEMANTiCS 2017 workshops proceedings: SIS-IoT
September 11-14, 2017, Amsterdam, Netherlands

[1] The ITEA2 SEAS project involved 34 partners in 6 countries, and ended in December 2016. It received the *ITEA2 Award of Excellence 2017*. See http://the-smart-energy.com/

2 COMBINING BEST PRACTICES, MODULARITY, VERSIONING, AND A UNIQUE NAMESPACE.

2.1 Best Practices

Design requirement 1. *IRIs must satisfy the following best practices defined in [3], [1], and [13].*

Thus: (i) IRIs must not change, and be designed with simplicity, stability and manageability in mind [3]; (ii) The description of a concept should be accessible by looking up its IRI [1, item 3]; (iii) Different representations of this description should be served depending on who asks [3]. For the stability and manageability aspect we use the w3id.org redirection service which is an initiative of the W3C and *is intended to be around for as long as the Web is around.*[2] Also, the sources of the SEAS website are open-source and available on GitHub: https://github.com/thesmartenergy/seas, so anyone can reuse, comment, raise issues, or contribute. Requirement 1 also imposes that resource IRIs must be either 303 IRIs or hash IRIs, because they are real-world objects.

Design requirement 2. *A reference ontology must be a valid OWL 2 DL ontology, satisfy the best practices for OWL documents as defined in [17, §3], and satisfy the Data on the Web best practices [5] and the Linked Open Vocabularies best practices [16].*

SEAS uses recommended metadata for the ontologies, most of the annotation properties are chosen in vocabulary `dct`, and term labels and comments systematically have language tags.

2.2 Modularization

Numerous use cases are envisioned for the IoT and they need knowledge representations means for very different domains (e.g., use cases in the SEAS project included Smart Homes, Micro Grids, Electric Vehicles, Electricity market, Distribution and Retail operators and clients, Weather Forecast). Not every part of an IoT reference ontology will be needed by every implementation. The SEAS ontology hence consists of a set different ontology modules. The choice of the modules is partly driven by a study of use cases and methodological principle [2].

Design requirement 3. *A reference ontology for the IoT must be modular using the mechanism described in [17, §3.4].*

We also chose to annotate every concept with a rdfs:isDefinedBy, that points to the ontology that defines the concept. Note that in this paper all the prefixes are those available with the service http://prefix.cc/.

The Semantic Web philosophy also encourages the reuse of existing ontologies when appropriate. The SEAS Use Cases required models that are common to numerous other projects (e.g., Provenance, Time instants and intervals, Quantities and Units of Measure, Time, Sensor/Actuator description, Sensor Observations, Predictions), or that are not actual subdomains of the Energy domain (e.g., products and offers). There exists some ontologies for some of these domains, and we chose to import or to define alignments to them on a per-ontology basis. The SEAS ontologies only imports OWL-Time [4] and GoodRelations [8], and defines alignments to SSN [7] and QUDT [10]. These alignments are defined in external modules, that import both the SEAS ontologies and the existing ontology. One can hence choose between using SEAS only, or SEAS in conjunction with the existing ontology.

[2]Thus longer than the website of most institutions or companies, hopefully.

2.3 Versioning

Much like softwares, ontologies evolve. Another ontology, some dataset, or a piece of software that uses some ontology must not be broken whenever it evolves.

Design requirement 4. *A reference ontology for the IoT must use a versioning mechanism, such as the one defined in [17, §3.3].*

This includes the annotation of the ontology using owl:versionIRI, owl:versionInfo, and potentially owl:priorVersion that points to the URI of the previous module version. An ontology series O is identified by an IRI, and each of the ontology versions O_i are also identified by IRIs. Design requirement 1 imposes that: An ontology version O_i identified by IRI v_i in an ontology series O identified by IRI u SHOULD be accessible via the IRI v_i. Furthermore, if O_i is the latest version of the ontology series O, then it SHOULD also be accessible via IRI u. Then the version numbers must be meaningful:

Design requirement 5. *A reference ontology for the IoT must use some adaptation of the Semantic Versioning specification 2.0+ to the linked vocabularies [12], as already discussed in [9].*

An ontology document is a document, so one may return an HTTP code 200 OK when serving it. For instance, seas:BuildingOntology has two versions: seas:BuildingOntology-0.9 and seas:BuildingOntology-1.0; the latter being the current latest version.

2.4 A unique Namespace

There are several reasons why industrial partners wanted to have a unique namespace for every term in the ontology. The most important ones where: (i) one does not want to have to remember a different randomly chosen prefixes for every term; (ii) one wants to be able to move a term from one module to another without having to change its URI; and (iii) the objective of SEAS is to grow with more and more terms, and to be able to let users define their own modules on the basis of a coherent set of terms. We hence decided to work with a unique namespace:

Design requirement 6. *All of the resources IRIs in the SEAS ontology must be in the same namespace.*

This solution raises one minor issue in the Linked Open Vocabulary (LOV) ontology repository [15]. the LOV has been designed with the assumption of 1:1 correspondences between prefixes and vocabularies. So it fails at properly referencing the terms that are defined in the SEAS ontologies: the SEAS vocabulary consists of several ontologies that have different IRIs but the same prefix.

2.5 Resolution

Let us show how one can combine requirements 1 to 6. Suppose an ontology O with IRI `OI` defines the class of "systems" R.

Hash IRIs. Let `RI#fragment` be a hash IRI for R. Operating a HTTP GET to `RI` must return a document with HTTP code 200 (this satisfies requirement 1). The returned document should define R. Hence

`OI` must be equal to `RI`. For instance, `RI` = https://w3id.org/seas/system#System. Suppose another resource R_2 in another ontology O_2 has a IRI `OI2#fragment2`. Requirement 6 imposes that string 'seas:' be the prefix for both namespaces `OI#` and `OI2#`. Which is impossible.

The solution is 303 IRIs. Let `RI` be a 303 IRI for R. Operating a HTTP GET to `RI` must return a HTTP code 303 See Also that redirects to a IRI `RI2`. The definition of R should be accessible via `RI2`.

Ontology O defines R. Hence we propose that `RI` redirects to `OI`. For instance, `RI` = https://w3id.org/seas/System redirects to `OI` = https://w3id.org/seas/SystemOntology. Hence Requirement 6 is satisfied and there exists a unique prefix for 'seas', whose extended version is `<https://w3id.org/seas/>`. This namespace is registered at the well known service http://prefix.cc/seas.

To sum up, when looking up a IRI `RI`: (1) If `RI` identifies an ontology series O, then the ontology server redirects to the latest version of this series with HTTP code 303 See Other; (2) If `RI` identifies an ontology document O, then the ontology server returns O with HTTP code 200 OK; (3) If `RI` identifies a resource R, then the ontology server 303 redirects to the ontology module with the most recent version that defines R with HTTP code 303 See Other. (4) each module version is available in the Turtle, RDF/XML, and HTML formats, with server content negotiation, reference to a unique canonical URI for each representation, and hint for a filename to use if the browser wants to download the file. For example, operating a HTTP GET to https://w3id.org/seas/EvaluationOntology with `Accept: text/turtle`, one gets an HTTP response with the following headers:

```
Content-Disposition: filename= EvaluationOntology-1.0.ttl;
Content-Location: https://w3id.org/seas/EvaluationOntology-1.0.ttl
```

3 OVERVIEW OF THE SEAS REFERENCE ONTOLOGY

The SEAS knowledge model is a modularized and versioned ontology with a core of four modules that describe among other physical systems and their connections, value association for their properties, and the activities by which such value association is done.

On top of these core modules, the SEAS ontology defines several "vertical" modules, that instantiate one or more core ontology patterns, and usually depend on a domain. For example:

- seas:ElectricPowerSystemOntology defines system types (e.g., seas:ElectricPowerLine, seas:ElectricPowerProducer), their connections through which they exchange energy (e.g., seas:SplitPhasePowerBus), their properties (e.g., seas:power, seas:voltage), and the classical types of evaluation for these properties (e.g., seas:NominalOperatingEvaluation).
- seas:ZoneOntology defines physical zones and their frontiers, through which these zones can exchange agents, thermic energy, light, or humidity.

Other vertical modules are proposed for energy markets, demand/response, etc. Labels and comments for the concepts are available at least in English, but sometimes in French, Finnish, and/or Portugese. The ontologies use concepts from external vocabularies.

When loading the SEAS ontologies in the Protégé editor from their IRI, and removing imports to external ontologies (OWL-Time and GoodRelations), one currently obtains the following metrics: 1873 logical axioms, 479 classes, 277 object properties, 7 data properties, 52 individuals. The modules and how they import each other is illustrated on figure 1.

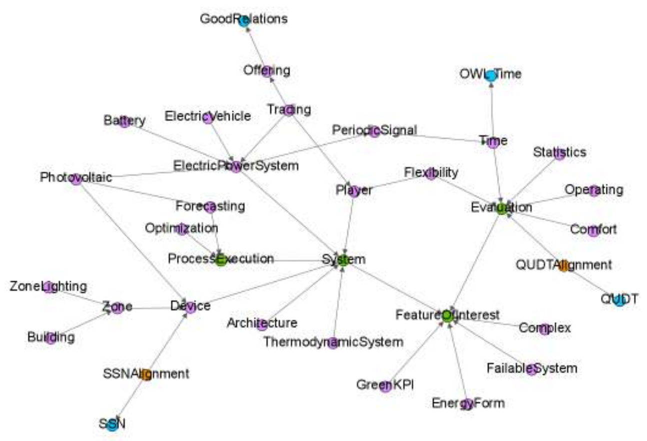

Figure 1: Illustration of the modules and their imports. Green nodes are core modules, pink nodes are vertical modules, orange nodes are alignment modules, and blue nodes are external ontologies.

These ontologies are published in Turtle, XML, and HTML according to the Web of Linked Open Vocabularies best practices. Their documentation is automatically generated in HTML thanks to a tool we developed.

We are also currently reviewing pull requests for extension modules to model matter flow systems, and city lighting.

4 FEATURES OF INTEREST AND THEIR PROPERTIES

Module seas:FeatureOfInterestOntology borrows the core concepts from the SOSA/SSN ontology, and defines ontology patterns to describe features of interest (e.g., a light) and their properties (e.g., its on/off status), which are qualifiable, quantifiable, observable or actionnable qualities of the feature of interest. More specifically, a seas:FeatureOfInterest is *an abstraction of real world phenomena (thing, person, event, etc)*, and a seas:Property is *an observable or operable Quality of an Event or Object. That is, not a quality of an abstract entity as is also allowed by DUL's Quality, but rather an aspect of an entity that is intrinsic to and cannot exist without the entity and is observable by a sensor, or operable by an actuator.*

4.1 Instances of the Ontology Pattern

One can then instantiate the ontology patterns and define for example in ontology `ex:` that `ex:Light` is a type of seas:FeatureOfInterest, `ex:BooleanProperty` is a type of seas:Property, and `ex:onStatus` is a subproperty of seas:hasProperty. One can furthermore state that property `ex:onStatus` is functional, meaning that features of interest can have only one on/off status property. Then, one can link a specific

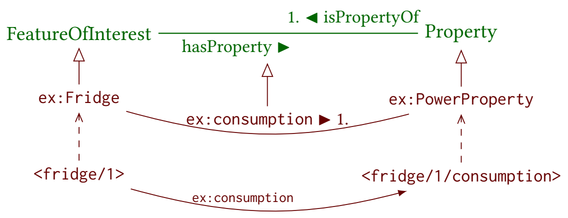

Figure 2: Ontology pattern seas:FeatureOfInterestOntology (in green), its instantiation (in ontology ex:) and some data.

light `<light/1>` to its (unique) on/off status property `<light/1/status>` using the following RDF graph:

```
<light/1> a ex:Light, seas:FeatureOfInterest ;
  ex:consumption <light/1/status> .
<light/1/consumption> a ex:BooleanProperty, seas:Property .
```

Differentiating in this way the subclasses of seas:Property and the subproperties of seas:hasProperty has two main advantages: (1) It dissociates what is a role with respect to a feature of interest (e.g., incoming and outgoing electric power), and what is intrinsic to those properties (e.g., they are seas:PowerProperty and have a physical dimension of *power*). (2) The functional aspect of some of the subproperties of seas:hasProperty ensures they are used consistently. For example a light has exactly one value for property on/off status in a given context.

The current version of SEAS contains 65 subclasses of seas:Property and 157 sub-properties of seas:hasProperty, among which 142 are functional. These properties cover (non-exhaustive list): (a) time-related properties (event number, state change frequency, state change number, state duration), (b) complex numbers (real, imaginary, module, phase); (c) periodic signals (frequency), (d) physical systems and areas (location, length, width, diameter, shape, speed, area, volume, humidity of various kinds); (e) population and population flow, noise, air pollution of various kinds; (f) power and energy (outgoing, incoming, produced, consumed, stored); (g) electric systems (resistance, conductance, reactance, susceptance, voltage, current); (h) failable systems (checkup, failure, repair, replace, unavailability); (i) thermodynamic systems (temperature, thermal transmittance, total heat transfer); (j) ecology (carbon footprint), (k) financial systems (price, tax, value added tax), (l) lighting (colour, light reflection and transmission, luminosity, refractive indices)

4.2 Alignment with other ontologies

4.2.1 Properties in SOSA/SSN. The definition of ssn:Property in the previous version of SSN imposed that an instance of ssn:Property is intrinsic to and cannot exist without a specific instance of seas:FeatureOfInterest. This is made clear in [7, §5.3.1.3.4]: *ObservedProperty is defined as a subclass of DUL:Quality. Types of properties, such as temperature or pressure should be added as subclasses of ObservedProperty instead of individuals.* The definition of seas:Property thus conforms to that of the original Semantic Sensor Network ontology. On the contrary, some of the referenced datasets that use SSN define types of properties such as temperature or pressure as instances of ssn:Property.[3] A specific section in the new SOSA/SSN ontology specification acknowledges the existence of these two different ways of modeling properties [7, §7.4], and discusses their compatibility.

On the other hand, the alignment to DUL in the old SSN ontology imposed a disjunction between ssn:FeatureOfInterest and ssn:Property. The absence of a normative DUL alignment in the new SSN relaxes this restriction, and enables some of the use cases that were envisioned in SEAS. For example, an alternating current power property has active, apparent, reactive power properties which have a quantity dimension of power, and some power factor property which is dimensionless.

4.2.2 Property in SAREF. SAREF also has a class that is named saref:Property. It is defined as *anything that can be sensed, measured or controlled in households, common public buildings or offices.* Although in the SAREF specification the concept *power* is used as an example of an instance of property,[4] we believe that saref:Property can be aligned to seas:Property (and thus ssn:Property, and benefit from the ontology patterns defined in this SEAS module.

4.2.3 Property in the WoT TD ontology. The current version of the WoT TD ontology [11] has been imported from the EU-H2020 VICINITY project and also contains a wot:Property class. This is a subclass of wot:InteractionPattern and has no overlap with ssn:Property, saref:Property or seas:Property.

5 PROCEDURES AND THEIR EXECUTIONS

The Procedure Execution ontology (PEP) defines pep:ProcedureExecutors (sensor, actuator, web service, etc.) that implement pep:Procedure methods (sensing, actuating, forecasting, some algorithm) and make pep:ProcedureExecution activities (observation, actuation, web service execution, forecast). pep:Procedures may be linked to some description of the input and/or the output using object properties pep:hasInput and pep:hasOutput. Their executions may be linked to some description of the command and/or the result using object properties pep:hasResult and pep:hasCommand. If the command or the result can be modeled as a simple RDF literal (a typed UNICODE string), then one may use datatype properties pep:hasSimpleResult and pep:hasSimpleCommand instead.

Figure 3 illustrates the Procedure Execution ontology ontology pattern in green.

As they represent very different concepts, classes pep:Procedure, pep:ProcedureExecutor, and pep:ProcedureExecution, are pairwise disjoint. Also, a pep:ProcedureExecution is made by at most one pep:ProcedureExecutor using at most one pep:Procedure. A Procedure has at most one input and at most one output, and a pep:Procedure has at most one command and at most one result. Finally, a procedure execution uses all of the procedures that are implemented by the executor that made it.

5.1 Instances of the Ontology Pattern

The procedure execution ontology is a simple generalization of the SOSA/SSN ontology. SOSA describes sosa:Sensors that implement sosa:Procedures and make sosa:Observations, which are

[3]See https://www.w3.org/2015/spatial/wiki/What_is_an_instance_of_ssn:Property

[4]In the definition of Unit of Measure: For example, Power is a property and Watt is a unit of power that represents a definite predetermined power

activities. In parallel to this, it describes sosa:Actuators that implement sosa:Procedures and make sosa:Actuations. The Procedure Execution ontology defines an ontology pattern as a generalization of these two parallel conceptual models, which accounts for at least a third use case: Web services exposed on the web may be called to trigger the execution of some procedure.

When instantiating the PEP pattern, one needs to define three classes, and potentially properties that link instances of these classes to some properties. Figure 3 illustrates such a pattern instantiation in some ontology `ex:` in blue.

SEAS currently contains several instances of the PEP ontology pattern, available in the following modules: (a) seas:DeviceOntology (*actuating* and *sensing*); (b) seas:ForecastingOntology (forecasting, forecaster, forecast); (c) seas:OptimizationOntology (optimization process, executor, and execution); (d) seas:TradingOntology (*trading* and *balancing*); (e) seas:SmartMeterOntology (*metering*, *consumption metering*, and more).

The seas:SmartMeterOntology and the different concepts it introduces such as seas:ConsumptionMetering and seas:ProductionMetering illustrate the fact that combinations can be created between ontology patterns: given a set of n instances of the properties ontology pattern and a set of m instances of the PEP ontology pattern, one can define $n \times m$ instances of a new ontology pattern that combines the two. We are currently investigating methods to ensure instances of such pattern combinations can be generated automatically, and means to filter out those that would be irrelevant.

5.2 Alignment to other ontologies

5.2.1 SOSA/SSN. The alignment between PEP and SOSA/SSN is defined in an external document pep:SSNAlignment. The naming of classes and properties in the PEP module has been changed to reflect the latest SOSA/SSN developments. In particular, new version 1.1: (i) defines pep:hasResult and pep:hasSimpleResult; (ii) uses term "made" instead of "executed"; (iii) uses term "procedure" instead of "process". On the other hand, SOSA defines only the input, output, result and simple result. PEP introduces two additional concepts for commands and simple commands. Change from PEP v1.0 to PEP v1.1 are still to be propagated to the other SEAS modules that imported PEP v1.0. On the other hand, nothing broke with this change of version because the ontology series and the individual ontology versions are clearly identified and kept published online. This illustrates the robustness of an ontology that adopts the quality criterions defined in section 2.5.

5.2.2 SAREF. Both classes saref:Function (i.e., functionalities that devices can perform to accomplish their tasks) and saref:Service (i.e., representation of these functions offered by the device over an IT network), along with their hierarchies (e.g., saref:SensingFunction, saref:ActuatingFunction, saref:EventFunction, saref:MeteringFunction, saref:SwitchOnService) may be used as subclasses of the seas:Procedure class. SAREF also proposes the class saref:Command with a set of predefined instances (saref:onCommand saref:offCommand) that could be used both as input description of a procedure, and as command of its execution.

5.2.3 The WoT TD ontology. The following alignments can be proposed between PEP and the WoT ontology:

```
wot:providesInteractionPattern rdfs:subPropertyOf pep:implements .
wot:InteractionPattern rdfs:subClassOf pep:Procedure .
wot:hasInputData rdfs:subPropertyOf pep:hasInput .
wot:hasOutputData rdfs:subPropertyOf pep:hasOutput .
```

5.2.4 RDFP. The seas:ProcedureExecution ontology is designed to be aligned with the RDFP ontology[5], which can be used to describe how a request/response message can be validated, lifted to RDF, or generated from a RDF representation of the request/response.

6 SYSTEMS AND THEIR CONNECTIONS

The seas:SystemOntology module defines an ontology pattern to describe systems that are connected via some connection points. This ontology pattern can be instantiated for example to describe zones inside a building (systems), that share a frontier (connections).

6.1 Systems

A system, modeled by class seas:System, is defined as a part of the universe that is virtually isolated from the environment. The system properties are typically state variables (e.g., consumed or stored energy, agent population, temperature, volume, humidity). Figure 4 illustrates classes and properties that can be used to define connected systems and their sub-systems.

A system may be connected to other systems that are part of its environment. This is modeled by property seas:connectedTo, which is symmetric. For example,

```
<electric_vehicle> seas:connectedTo
     <electric_vehicle_service_equipment> .
```

Connected systems interact in some ways. The exact meaning of *interact* is defined by sub properties of seas:connectedTo. For example, for the electricity to directly flow between an electric vehicle service equipment `<electric_vehicle_service_equipment>` and an electric vehicle `<electric_vehicle>`, then they must be linked by property seas:exchangesElectricityWith:

```
<electric_vehicle> seas:exchangesElectricityWith
     <electric_vehicle_service_equipment> .
```

A system can be a sub-system of a unique other system. This is modeled using property seas:subSystemOf, which is asymmetric and functional. For example,

```
<battery> seas:subSystemOf <electric_vehicle> .
```

Properties of subsystems somehow contribute to the properties of the super system. The exact meaning of this *contribution* is defined by sub properties of seas:subSystemOf. For example, if one wants to model the fact that the consumption power of a fridge `<fridge/1>` contributes to the consumption power of the kitchen, `<kitchen/1>`, then one may use a sub-property of seas:subSystemOf named seas:subElectricPowerSystemOf:

```
<fridge/1> seas:subElectricPowerSystemOf <kitchen/1> .
```

Property seas:subSystemOf is functional, and should be asymmetric. In fact, this would prevent a system from being its own sub-system. But OWL 2 DL prevents a non-simple property (e.g., a functional property) from being asymmetric, see [17, §11]. If it was

[5] RDF Presentation Ontology - https://w3id.org/rdfp/

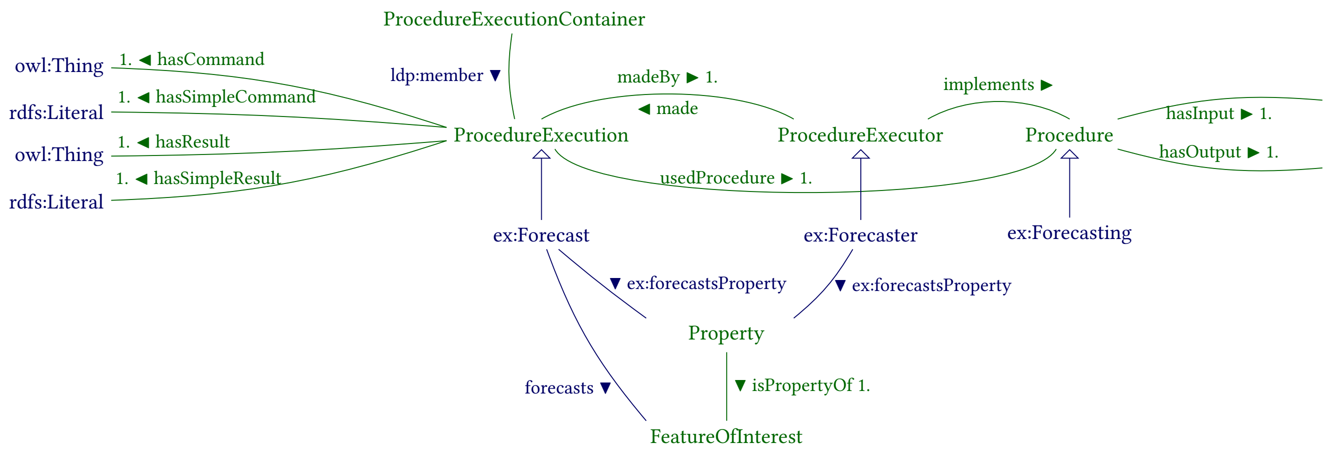

Figure 3: The Procedure Execution ontology ontology pattern and a possible instantiation.

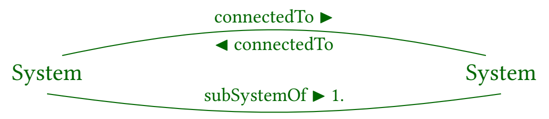

Figure 4: Module seas:SystemOntology: connected systems and sub-systems.

possible that both the fridge and the kitchen were sub-systems of a common super system, say, the house, then the consumption power of the fridge would contribute twice to the consumption power of the house. The functional aspect of property seas:subSystemOf prevents this undesired effect. Due to the open world assumption of RDF, it is not possible to model the closed set of sub systems of a system using property seas:subSystemOf.

6.2 Connections

A connection between two systems, modeled by seas:connectedTo, describes the potential interactions between connected systems. A connection can be qualified using class seas:Connection. For example, one can then associate this connection with properties that describe the interactions between the connected systems (e.g., population flow, exchange surface, contact temperature). Figure 5 illustrates classes and properties that can be used to qualify connections between systems.

```
<connection> seas:connectsSystem <electric_vehicle> ,
      <electric_vehicle_service_equipment> .
<electric_vehicle> seas:connectedThrough <connection> .
<electric_vehicle_service_equipment> seas:connectedThrough
      <connection> .
```

6.3 Connection points

A system connects to other systems at connection points. A connection point belongs to one and only one system, and can be described using the class seas:ConnectionPoint. Figure 6 illustrates the classes and the properties that can be used to describe connection points of a system.

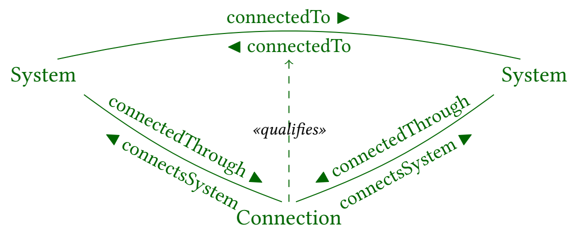

Figure 5: Module seas:SystemOntology: connections between systems.

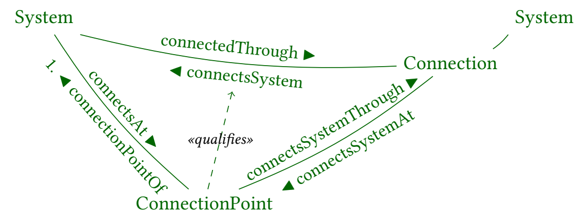

Figure 6: Module seas:SystemOntology: connection points of a system.

For example, an electric vehicle charging station may have three connection points: two plugs of different kind to which electric vehicles can connect, and a three phase connection point to the public grid:

```
<electric_vehicle> seas:connectsAt <plug_high_voltage> ,
      <normal_plug> , <three_phase_connection_point> .
```

One can then associate a connection point with properties that describe it (e.g., position and speed, voltage and intensity, thermic transmission coefficient).

6.4 Instances of the Ontology Pattern

This ontology pattern can be instantiated for different domains. For example to describe zones inside a building (systems), that share a frontier (connections). Properties of systems are typically state variables (e.g., agent population, temperature), whereas properties of connections are typically flows (e.g., heat flow).

The current version of the SEAS ontology contains 173 subclasses of seas:System, 42 subclasses of seas:Connection, and 32 subclasses of seas:ConnectionPoint. The use cases that are covered are: (a) seas:ThermodynamicSystemOntology defines thermodynamic systems that can exchange heat with one another, and that have sub-thermodynamic systems. (b) seas:CommunicationOntology defines communication devices that can have multiple communication connection points with various protocols, and exchange information when they are connected through a communication connection; (c) seas:ArchitectureOntology models the concepts from the SRAM SEAS Reference Architecture Model [6]: an extension of SmartM2M that allows for the distribution of group managers. This module defines group managers that can directly manage field entities, and core entities that can directly access field entities. (d) seas:ZoneLightingOntology defines light sources that can illuminate illuminable zones, which can then transmit light to other illuminable zones. (e) seas:TradingOntology defines electricity traders that can trade electricity *with* one another, or trade electricity *on* various electricity markets. (f) seas:ElectricPowerSystemOntology defines electric power systems that can exchange electricity with other electric power systems via electric power connection points. This module is the biggest and most detailed SEAS module. It has been developed in collaboration with researchers from GECAD/ISEP in August 2016, and allows to describe precisely the typology of electric grids, the electric power systems that compose it (transmission systems, transformers, producers, consumers, storage systems), and the configuration in which they are connected via Power buses they can share that depends on the current type of the connected system.

7 PROPERTY EVALUATIONS

Module seas:EvaluationOntology defines ontology patterns to describe evaluations (qualification or quantification) of properties, and to qualify these evaluations: (i) what kind of evaluation: (e.g., maximum tolerable temperature, forecasted temperature, mean temperature), (ii) in a validity context (e.g, between 10 am and 12 am tomorrow).

7.1 Direct evaluation

A seas:Property may be directly associated with a quality or quantity value, which is then unique and constant. This association is asserted using object property seas:value or using datatype property seas:simpleValue.

A quantity value may use external vocabularies such as QUDT (it would then be qudt:Quantity), or OM (it would then be a om:Measure or om:Point), or be directly encoded as a literal using an appropriate datatype such as cdt:ucum. For example, the following triples quantify the consumption of a fridge using cdt:ucum literals:

```
<fridge/1/con/frequency> seas:simpleValue "50.1 Hz"^^cdt:ucum .
<fridge/1/con/voltage> seas:simpleValue "231 V"^^cdt:ucum .
<fridge/1/cons/voltageTensionPhase> seas:simpleValue "1.68
      RAD"^^cdt:ucum .
```

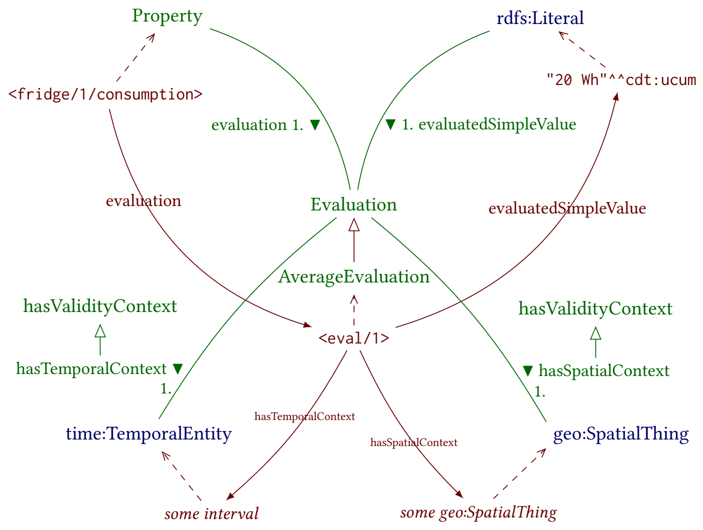

Figure 7: Illustration of module seas:EvaluationOntology and its use.

7.2 Quantified evaluations.

Because property values may evolve in space and time, or because they can be approximate measures or forecasts, class seas:Evaluation qualifies the link seas:value. In particular, an instance of seas:Evaluation may hold metadata about: (1) *The type of evaluation* is defined by the hierarchy of seas:Evaluation sub classes. SEAS currently contains 92 such classes, among which: (a) seas:MaximumComfortableEvaluation: the given value is the maximum value for a property that is considered comfortable for the associated agent; (b) seas:TimeMaximumEvaluation: the given value is the maximum of the quantity over the temporal context. (2) *The context of validity of the evaluation.* The W3C&OGC Spatial Data on the Web Working Group defined best practices for describing the validity context of entities [14, Best Practice 11]: *Spatial data should include metadata that allows a user to determine when it is valid for.* In SEAS, an evaluation validity context is described using functional sub properties of seas:hasValidityContext. This property can be aligned with SSN property ssn-system:inCondition, which *describes the prevailing environmental Conditions for SystemCapabilites, OperatingRanges and SurvivalRanges.* We defined two such properties as shown in Figure 7: (a) seas:hasTemporalContext links an entity to its temporal validity context, a time:TemporalEntity; (b) seas:hasSpatialContext links an entity to its spatial validity context, a geo:SpatialThing. (3) *Provenance information or any other data.* Other metadata may be added to describe an evaluation instance. For example the W3C PROV Ontology enables to describe the activity that generated the evaluation, or its generation time. Other vocabularies may be used to further describe evaluations.

For example, the following graph states that the maximum comfortable value of `<room/1/temp>` is 27.0 °C for agent `<Granny>`, but this value peaked at 35.8 °C during time interval 12:00-13:00.

```
<room/1/tempe> seas:evaluation [
  a seas:TemperatureEvaluation , seas:AgentComfortEvaluation,
   seas:MaximumComfortableEvaluation ;
  seas:relativeToAgent <Granny> ;
  seas:evaluatedSimpleValue "27.0 DEG"^^cdt:ucum ] ] .

<room/1/temp> seas:evaluation [
  a seas:TemperatureEvaluation , seas:TimeMaximumEvaluationn ;
  seas:evaluatedSimpleValue "35.8 DEG"^^cdt:ucum ;
  seas:hasTemporalContext [
   a time:Interval ;
   time:hasBeginning [ time:inXSDDateTimeStamp
     "2017-08-10T12:00:00Z"^^xsd:dateTimeStamp ] ;
   time:hasEnd [ time:inXSDDateTimeStamp
     "2017-08-10T13:00:00Z"^^xsd:dateTimeStamp ] ] ] .
```

7.3 Properties vs Evaluation: choice criteria

When properties seem similar and inter-dependent, one requires precise criteria to decide how to represent them best:

(A) *distinct properties.* If properties can be independent, one represent each of them distinctly, as functional sub-properties of property seas:hasProperty. This is how length and width are modeled for example. One can then give independent evaluation for each of these properties. (B) *a single property and different subclasses of seas:Evaluation.* If the property is unique, but one qualify/quantify it differently, then one only define a single functional sub-property of property seas:hasProperty, and several sub-classes of seas:Evaluation. This is how a length, and the evaluation of its minimum and maximum can be modeled. Note that the physical dimension given to a property must be the same as the physical dimension of each of its evaluation. (C) *a property and several properties of this property.* If one wants to model a unique property that itself has several properties with different quantity dimensions. This is how the consumed energy, with its active, reactive, apparent, and the power factor, are modeled.

Using OWL axioms (or rules), one may express an equivalence between options (A) and (C) above.

8 CONCLUSION

In this paper we described the design and publication choices that were adopted in the development of the SEAS ontology with the goal of making it a high-quality, modular and versioned, ontology, with a homogeneous, predictible, and extensible structures thanks to the simple core SEAS ontology patterns and the way they can be instantiated. The coverage of SEAS can be extended for the modeling and the description of any kind of engineering-related data/information/systems, without giving up on any of the recommended best practice def ined by the Semantic Web community. Together with the SAREF and the ETSI SmartM2M momentum, SEAS represents a potential additional step towards large adoption of the Semantic Web formalisms by industry stakeholders of multiple verticals, meaning a step closer to actual semantic interoperability on the IoT.

Future work on the SEAS content include extensions for multiple engineering-related verticals such as Smart Home, Smart Building, Electric Mobility, Industry of the Future/Industry 4.0, including all their field devices/processes/systems, measurements, environment, actors/players and their relations, as well as flexibility/trading/business related aspects.

Future work on the SEAS development ecosystem includes development of continuous integration tools that automatically check the quality of pull requests on gitHub; declarative description of ontology patterns instantiations and compound ontology pattern instantiation with filters (ongoing work), and an enhanced documentation and publication platform for the ontology with enhanced management of user-requested terms and modules.

ACKNOWLEDGMENTS

This paper has been partly financed by the ITEA2 12004 SEAS (Smart Energy Aware Systems) project, the ANR 14-CE24-0029 OpenSensingCity project, and a bilateral research convention with ENGIE R&D. The author would like to thank animators and participants of the first SEAS knowledge engineering workshop that kicked-off this work, and especially domain experts from ENGIE R&D, ISEP/GECAD, CNR, ARMINES, IMT, INNOVA, VTT, ASEMA.

REFERENCES

[1] 2005. Linked Data. Published online at http://www.w3.org/DesignIssues/LinkedData.html. (2005). W3C Design issue.
[2] Jie Bao, George Voutsadakis, Giora Slutzki, and Vasant G. Honavar. 2009. Package-Based Description Logics. In *Modular Ontologies: Concepts, Theories and Techniques for Knowledge Modularization*, Heiner Stuckenschmidt, Christine Parent, and Stefano Spaccapietra (Eds.). Lecture Notes in Computer Science, Vol. 5445. Springer-Verlag, 349–371.
[3] Tim Berners-Lee. 1998. *Cool URIs don't change.* W3C Note. W3C. https://www.w3.org/Provider/Style/URI.html
[4] Simon Cox and Chris Little. 2017. *Time Ontology in OWL.* W3C Candidate Recommendation. W3C. https://www.w3.org/TR/2017/CR-owl-time-20170606/
[5] Bernadette Farias Lóscio, Caroline Burle, and Newton Calegari. 2017. *Data on the Web Best Practices.* W3C Recommendation. W3C.
[6] Guillaume Habault, Jani Hursti, and Jean-Marie Bonnin. 2017. Defining a Distributed Architecture for Smart Energy Aware Systems. In *Complex Systems Design & Management.* Springer, 83–94.
[7] Armin Haller, Krzysztof Janowicz, Simon Cox, Danh Le Phuoc, Kerry Taylor, and Maxime Lefrançois. 2017. *Semantic Sensor Network Ontology.* W3C Candidate Recommendation. W3C. https://www.w3.org/TR/2017/CR-vocab-ssn-20170711/
[8] Martin Hepp. 2008. Goodrelations: An ontology for describing products and services offers on the web. *Knowledge Engineering: Practice and Patterns*, 329–346.
[9] Diane I Hillmann, Gordon Dunsire, and Jon Phipps. 2014. Versioning vocabularies in a linked data world. (2014).
[10] Ralph Hodgson, Paul J Keller, Jack Hodges, and Jack Spivak. 2014. QUDT-quantities, units, dimensions and data types ontologies. (2014). USA, Available from: http://qudt.org [March 2014].
[11] Sebastian Kaebisch and Takuki Kamiya. 2017. *Web of Things (WoT) Thing Description.* W3C First Public Working Draft. W3C. https://www.w3.org/TR/wot-thing-description/
[12] Preston-Werner, Tom. 2013. *The Semantic Versioning specification.* Technical Report. http://semver.org/
[13] Leo Sauermann and Richard Cyganiak. 2008. *Cool URIs for the Semantic Web.* W3C Note. W3C. https://www.w3.org/TR/cooluris/
[14] Jeremy Tandy, Linda van den Brink, and Payam Barnaghi. 2017. *Spatial Data on the Web Best Practices.* W3C Working Group Note. W3C. https://www.w3.org/TR/2017/NOTE-sdw-bp-20170511/
[15] Pierre-Yves Vandenbussche, Ghislain A Atemezing, María Poveda-Villalón, and Bernard Vatant. 2017. Linked Open Vocabularies (LOV): a gateway to reusable semantic vocabularies on the Web. *Semantic Web* 8, 3 (2017), 437–452.
[16] Pierre-Yves Vandenbussche and Bernard Vatant. 2012. *Metadata recommendations for linked open data vocabularies.* Web document. https://lov.okfn.org/Recommendations_Vocabulary_Design.pdf
[17] W3C OWL Working Group. 2012. *OWL 2 Web Ontology Language Structural Specification and Functional-Style Syntax (Second Edition), W3C Recommendation 11 December 2012.* Technical Report. W3C. http://www.w3.org/TR/2012/REC-owl2-syntax-20121211/