SOSA: A Lightweight Ontology for Sensors, Observations, Samples, and Actuators

Krzysztof Janowicz[a], Armin Haller[b], Simon J D Cox[c], Danh Le Phuoc[d], Maxime Lefrançois[e]

[a]*Geography Department, University of California, Santa Barbara, USA*
[b]*Research School of Computer Science, Australian National University, Canberra, Australia*
[c]*Land and Water, CSIRO, Melbourne, Australia*
[d]*Department of Telecommunication Systems, Technische Universität, Berlin, Germany*
[e]*Connected-Intelligence team, Ecole des Mines de Saint-Etienne, France*

Abstract

The Sensor, Observation, Sample, and Actuator (SOSA) ontology provides a formal but lightweight general-purpose specification for modeling the interaction between the entities involved in the acts of observation, actuation, and sampling. SOSA is the result of rethinking the W3C-XG Semantic Sensor Network (SSN) ontology based on changes in scope and target audience, technical developments, and lessons learned over the past years. SOSA also acts as a replacement of SSN's Stimulus Sensor Observation (SSO) core. It has been developed by the first joint working group of the Open Geospatial Consortium (OGC) and the World Wide Web Consortium (W3C) on *Spatial Data on the Web*. In this work, we motivate the need for SOSA, provide an overview of the main classes and properties, and briefly discuss its integration with the new release of the SSN ontology as well as various other alignments to specifications such as OGC's Observations and Measurements (O&M), Dolce-Ultralite (DUL), and other prominent ontologies. We will also touch upon common modeling problems and application areas related to publishing and searching observation, sampling, and actuation data on the Web. The SOSA ontology and standard can be accessed at `https://www.w3.org/TR/vocab-ssn/`.

Keywords: Ontology, Sensor, Observation, Actuator, Linked Data, Web of Things, Internet of Things, Schema.org

1. Introduction and Motivation

In their broadest definition sensors detect and react to changes in the environment that directly or indirectly reveal the value of a property. The process of determining this, not necessarily numeric, value is called an observation. Observation procedures provide a sequence of instructions to ensure that the observations are *reproducible* and *representative*, whereby an individual assessment characterizes a feature (i.e., entity) of interest. Typically, observations are not carried out on the entire feature but on samples of it, or on an immediately sensed spatiotemporal region. The process of sampling may itself be specified by a procedure that determines how to obtain samples. Some observation procedures can contain sampling procedures as their parts. Actions triggered by observations are called actuations and the entities that perform them are actuators. Finally, actuators, sensors, and samplers are typically mounted on a platform. These platforms serve a wide range of needs, including carrying systems along a defined trajectory, protecting them from external influences that may distort the results, or spatially positioning multiple systems following a particular layout.

In the context of smart homes, for instance, a temperature sensor can be mounted to a wall and take repeated observations at some time interval. Each of these observations returns the temperature of a sample, namely the surrounding body of air. In cases where the sensor is placed correctly, the temperature is said to be characteristic (representative) for the entire feature of interest, e.g., a bedroom. For example, a decrease in room temperature may trigger an actuator to close the windows. While each individual observation results in a new value and is taken from a new sample, all observations are based on the same procedure, observe the same property, and reveal the same characteristic of the same feature of interest. Ignoring some aspect of the observation procedure - e.g., by placing the sensor next to the window so that the sampled body of air is no longer a suitable proxy for the entire room - may cause the observation results to become unrepresentative of the room's temperature, and may lead to the actuator closing the window unexpectedly. Note that if one would place all sensors in such way, the resulting observations may not be representative for the room though they still may be reproducible. Finally, if some sensors would be placed near windows while others would not, it would no longer be possible to establish a relationship between the behavior of individual actuators. Finally, procedures are specific for certain types of observations. Hence, one can follow a specific procedure and thereby arrive at reproducible results that is not representative or suitable for the task at hand.

With a rapid increase in data from sensors being published on the Web, there is an increasing interest in the re-use and combination of that data. However, raw observation results do not provide the context required to interpret them properly and

Email addresses: `janowicz@ucsb.edu` (Krzysztof Janowicz), `armin.haller@anu.edu.au` (Armin Haller), `simon.cox@csiro.au` (Simon J D Cox), `danh.lephuoc@tu-berlin.de` (Danh Le Phuoc), `maxime.Lefrancois@emse.fr` (Maxime Lefrançois)

to make sense of these data. Searching, reusing, integrating, and interpreting data requires more information about the studied feature of interest, such as a room, the observed property, such as temperature, the utilized sampling strategy, such as the specific locations and times at which the temperature was measured, and a variety of other information. With the rise of smart cities and smart homes as well as the Web of Things more generally, actuators and the data that is produced by their in-built sensors also become first-class citizens of the Web. Given their close relation to sensors, observations, procedures, and features of interest, outlined above it is desirable to provide a common framework and vocabulary that also includes actuators and actuation. Finally, with today's diversity of data and data providers, notions that restrict the view of sensors to being technical devices need to be broadened. One example would be social sensing techniques such as semantic signatures [14] to study humans and the data traces they actively and passively emit from within a sensor-observation framework. Simulations and forecasts are other examples showcasing why 'sensors' that produce estimates of properties in the world are not necessarily physical entities.

The Sensor Web Enablement standards such as the Observations and Measurements (O&M) [5] model and the Sensor Model Language (SensorML) [2] specified by the Open Geospatial Consortium (OGC) provide means to annotate sensors and their observations. However, these standards are not integrated and aligned with Semantic Web technologies, Linked Data, and other parts of the World Wide Web Consortium's (W3C) technology stack that aims at creating and maintaining a global and densely interconnected graph of data. The W3C Semantic Sensor Network Incubator Group (SSN-XG) tried to address this issue by first surveying the landscape of semantically-enabled sensor specifications [4] and then developing the Semantic Sensor Network (SSN) ontology [3] as a human and machine readable specification that covers networks of sensors and their deployment on top of sensors and observations. To provide an axiomatization beyond mere surface semantics, SSN made use of the foundational Dolce UltraLight (DUL) ontology, e.g., to state that platforms are physical objects. At the same time, SSN also provided the Sensor-Stimulus-Observation (SSO) [16] ontology design pattern [7] as a simple core vocabulary targeted towards lightweight applications and reuse-by-extension.

The broad success of the initial SSN led to a follow-up standardization process by the first joint working group of the OGC and the W3C. One of the tasks of this *Spatial Data on the Web* working group was to rework the SSN ontology based on the lessons learned over the past years and more specifically to address changes in scope and audience, shortcomings of the initial SSN, as well as technical developments and trends in relevant communities. The resulting ontology, published as a W3C Recommendation and OGC Standard [11], is not only an update but has been re-envisioned completely from the beginning. Most notably, and as depicted in Fig. 1 the revised ontology is based on a novel modular design which introduces a horizontal and vertical segmentation. Vertical modules add additional depth to the axiomatization, while horizontal modules broaden the ontology's scope, e.g., by introducing classes and relations to specify system capabilities or sample relationships. The modularization addresses an often voiced concern about the initial SSN release, that the DUL alignment introduced too strong ontological commitments, and the full ontology was too heavyweight for smart devices in the context of the Web of Things, and was running against the trend towards lightweight vocabularies preferred by the Linked Data and Schema.org communities. The proposed modularization allowed us to keep the DUL alignment for those who want to use it, and introduce additional alignments to Prov-O [20], O&M [5], and OBOE [23], while keeping the overall target audience broad, ranging from web developers and scientists that want to publish their data on the Web, to Web of Things industry players.

Figure 1: SOSA and its vertical and horizontal modules with the arcs indicating the direction of the import statement. Horizontal modularization is shown by arcuate modules at the same radius while vertical modularization is shown by modules at a larger radius.

The resulting collection of modules, including SSN, all build upon a common core: the **S**ensor, **O**bservation, **S**ample, and **A**ctuator ontology (SOSA). SOSA does not merely replace the former SSO ontology design pattern but provides a flexible yet coherent framework for representing the entities, relations, and activities involved in sensing, sampling, and actuation. It is indented to be used as a lightweight, easy to use, and highly extendable vocabulary that appeals to a broad audience beyond the Semantic Web community but can be combined with other ontologies, such as SSN to provide a more rigorous axiomatization where needed. At the same time, SOSA acts as minimal interoperability fall-back level, i.e., it defines those common classes and properties for which data can be safely exchanged across all uses of SSN, its modules, and SOSA.

In the following, we will focus on providing an overview of the main classes and properties of SOSA. We will also briefly

discuss their integration with the new SSN ontology. We will motivate some of the core design decisions and provide modeling examples that will arise in practice. For the sake of readability, we will focus on an examples-driven description of these classes. The formal and normative SOSA ontology and standard can be accessed at `https://www.w3.org/TR/vocab-ssn/`. Finally, we will discuss selected modeling problems and how to approach them and will give examples for the usage of SOSA classes and relationships in different application areas.

2. SOSA in a Nutshell

Here, we will highlight the most important classes and relationships that make up the SOSA ontology. In contrast to the original SSN, SOSA takes an event-centric perspective and revolves around observations, sampling, actuations, and procedures. The last is a set of instructions specifying how to carry out one of the three aforementioned acts. This event-centric modelling is aligned with community expectations, in particular the schema.org community that only cares about the digital representation of an event (i.e. Action class in the schema.org model) [9] and not the real-world process that underlies such an event. Modelling events as first-class citizens is also aligned with the PROV-O model [20] and standardized business process models that are used for Web service interfaces [28].

Fig. 2 depicts the SOSA ontology design pattern, called *core structure*, underlying all of the three modeling perspectives. The *activities* of observing, sampling, and actuating, each target some *feature of interest* by either changing its state or revealing its properties, each follows some *procedure*, and each is carried out by some object or *agent*. The core structure aligns well with other activity-centric standard ontologies such as the OGC Observations and Measurements [1] (i.e. `Observation` ≡ `so19156-om:OM_Observation`) and the PROV Ontology [20] (i.e. `Observation` ⊑ `prov:Activity`). Fig. 2 shows how some of the core classes in SOSA relate to an activity-centric model such as PROV-O. Note that SOSA does not model the ultimate agent that triggered an act, e.g., a person taking a reading off a sensor. We will show an example how this agent relationship can be modelled by using PROV-O in combination with SOSA in Sec. 4.2.

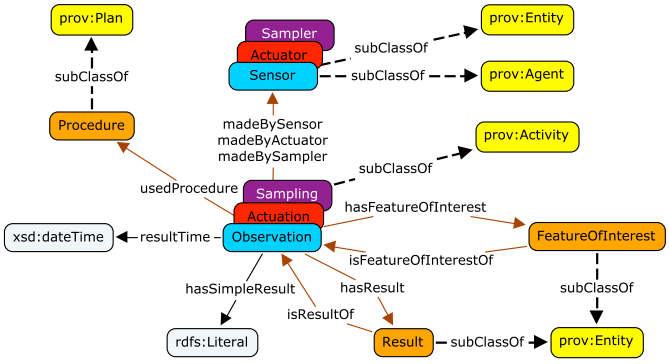

Figure 2: The core structure of SOSA, showing relationship to Prov-O

SOSA aims to strike a balance between the expressivity of the underlying description logic, the ease of use of language features, e.g., as measured by understanding their implications, cf. [27, 13], and the expectations of the target audience (including web developers and domain scientists), while accommodating a broad range of domains and application areas. Given SOSA's axiomatization, the resulting DL language fragment is $\mathcal{ALI}(\mathcal{D})$ which is efficiently supported by modern triple stores. In comparison the new SSN module's axiomatization results in $\mathcal{ALRIN}(\mathcal{D})$, while the old SSN was specified in $\mathcal{SRIQ}$.

To give a concrete example, SOSA does not declare classes to be disjoint despite this being among the most powerful language elements in terms of reasoning. This is for good reasons as it enables classes such as sensors and samples to be features of interest themselves. Consequently, one can make observations *using* sensors as well as *about* sensors. Similarly, features of interest can be regarded as the results of a sampling activity. Another notable design decision is the usage of Schema.org *domainIncludes* and *rangeIncludes* annotation properties [9] to provide an informal semantics, compared to the inferential semantics of their RDFS counterparts. The SSN vertical module continues to utilize *guarded* domain and range restrictions as part of its richer axiomatization.

2.1. Procedures

A `Procedure` is a workflow, protocol, plan, algorithm, or computational method that specifies how to carry out an observation, collect a sample, or make a change to the state of the world via an actuator. A procedure is re-usable in the sense that it is executed whenever one performs observations, actuations or samplings of a certain type, thereby making the results reproducible. Hence, procedures, are also crucial for fostering semantic interoperability [14] as `Procedures` linked to observation and sampling activities are typically a record of how these activities are/were performed. Procedures linked to actuation activities, however, can either be a record of how the actuation has been performed or a description of how to interact with an actuator (i.e., the recipe for performing actuations). The definition of the inputs and outputs of a procedure that define how to interact with an `Actuator` (i.e. its interface) are beyond the scope of SOSA and are relegated to SSN in combination with other standards/models such as the Linked Data Platform protocol [29] or the currently under development W3C Thing Description [17].

To give a concrete example of the modelling of `Procedures`, the measured wind speed differs depending on the height of a sensor above the surface, e.g., due to friction. Hence, procedures for measuring wind speed define a standard height for anemometers above ground, typically 10m for meteorological measures and 2m in Agrometeorology. This definition of height, sensor placement, avoidance of obstructions, etc., as expressed in the `<AgrometeorologyWindSpeed>` procedure in the example in Listing 2.1 make the observation results reproducible and representative. Multiple successive observations will yield comparable results and the measured values will be representative for the feature of interest, e.g., wind speed at a certain

stretch of California's Central Coast. While the number of (observation) acts increases constantly, any change in procedure (e.g., an increase in the number of used procedures) would be unexpected and may point to a novel technique, theory, or instrument.

Listing 2.1: Reusable procedures

```
<AgrometeorologyWindSpeed> a sosa:Procedure ;
  rdfs:comment "Instructions for measuring wind speed 2m above
       ground surface for applications in Agrometeorology."@en ;
  [...].

<AWSObs1> a sosa:Observation ;
  [...]
  sosa:usedProcedure <AgrometeorologyWindSpeed> .

<AWSObs2> a sosa:Observation ;
  [...]
  sosa:usedProcedure <AgrometeorologyWindSpeed> .
```

2.2. *Sensors and Observations*

An `Observation` is the act of carrying out an (observation) `Procedure` in order to estimate or calculate a value of an `ObservableProperty` of a `FeatureOfInterest` or a `Sample` thereof. An observation involves a `Sensor` and yields a `Result`. While SOSA relies on QUDT [25] or other vocabularies to describe observation results and their values, an additional datatype property is provided to handle the simple case that merely requires a typed literal, via the `hasSimpleResult` property. The `hasResult` object property allows one to model `Results` as first class citizens and make statements about them by, for example, stating the unit of measurement for the value. Listing 2.4 shows an example of the use of the `hasSimpleResult` for an `Observation`, while Listing 2.6 models a `Result` object with a unit of measurement defined for the object.

Sensors in SOSA can be physical devices, but also simulations, numerical models or people, to give a few examples. Sensors respond to stimuli, e.g., a change in the environment, or input data composed from the results of prior observations, to generate the result. Conceptually, they can be thought of as implementations (of parts) of an observation procedure. One or more sensors (as well as actuators and samplers) can be hosted on a `Platform`. Such platforms can also define the geometric properties, i.e., placement, of sensors in relation to one another. SOSA does not provide explicit properties to model the geometry, but delegates this to ontologies dealing with space, metric and topological properties, e.g., GeoSPARQL [26]. SOSA also distinguishes between `phenomenonTime` and `resultTime`. The former is the time that the result of an act of observation, actuation, or sampling applies to the feature of interest, while the latter specifies the *instant* of time when the act was completed. Consequently, `resultTime` is a datatype property, while `phenomenonTime` - which may be an interval or an instant - is an object property which utilizes the OWL-Time vocabulary[1].

[1] `https://www.w3.org/TR/owl-time/`

Finally, observable properties are similar to procedures or units of measure in the sense that they are singletons; one observable property will apply to many acts of observation, concerning different features of interest, at different times, or using different procedures or sensors, e.g., `<NitrateConcentration>` of rainwater from different storms as in Listing 2.2. Hence, to foster interoperability and reproducibility, we expect observable properties (and procedures) to come from controlled vocabularies based on code lists typically used in the sciences, e.g., SDN:P01::RNCNIT09[2]. Note that the SOSA axiomatixation does not make any formal restrictions on how to use observable properties, thus allowing for alternative models. Features of interest are not restricted to objects such as trees but also include events, e.g., a storm. In fact, sensors themselves can be features of interest for other sensors. For instance, an ophthalmologist uses sensors to examine a human's eye. Fig. 3 depicts the discussed classes and their relations.

Listing 2.2: Reusable Observable Properties

```
<NitrateConcentration> a sosa:ObservableProperty .
<NCR_Ion_Chromatography_Procedure> a sosa:Procedure.
<MetrohmXYZIonChromatographySystems> a sosa:Sensor.

<NC_S1> a sosa:Observation ; [...]
 sosa:usedProcedure <AgrometeorologyWindSpeed> ;
 sosa:madeBySensor <MetrohmXYZIonChromatographySystems>;
 sosa:observedProperty <NitrateConcentration>.

<NC_S2> a sosa:Observation ; [...]
  sosa:observedProperty <NitrateConcentration> .
```

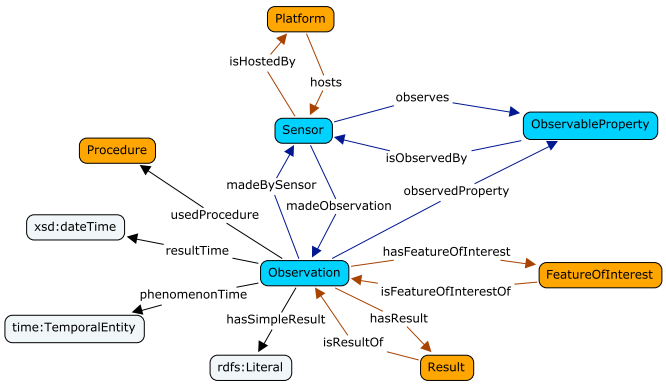

Figure 3: Overview of the SOSA Observation perspective

One or more sensors, actuators or samplers can be hosted or mounted on a `Platform`. Such platforms provide, for example, geometric properties essential for performing observations, actuation or sampling, e.g., by placing the microphone at a certain position and away from the speakers, the flash pointing in the same direction as the rear-facing camera, and so on. `Platforms` can also host other `Platforms`.

Increasingly, accelerometers, linear actuators, gyroscopes, barometers, magnetometers, microphones, cameras and other

[2] "Concentration of nitrate NO3- CAS 14797-55-8 per unit volume of rainwater [dissolved plus reactive particulate <0.4/0.45um phase] by ion chromatography"; URI `http://vocab.nerc.ac.uk/collection/P01/current/RNCNIT09/`.

sensors and actuators are mounted on modern smartphones, which can be modelled as platforms in SOSA. Listing 2.3 defines an IPhone 7 as the platform that hosts a Bosch Sensortec BMP282, a taptic engine linear actuator and a GPS sensor. The listing also shows how the location of the smartphone measured by the GPS sensor with its latitude and longitude (and expressed through the Geo Vocabulary [3]) at a given time can be modelled as an observable property to an observation. Other than to an observation, spatial location properties can also be attached to features of interest, platforms, sensors, actuators or samplers to allow both, the modelling of static and remote or mobile sensing/actuation/sampling.

Listing 2.3: Platform modelling

```
<iphone7/35-207306-844818-0> a sosa:Platform ;
  rdfs:label "IPhone 7 - IMEI 35-207306-844818-0"@en ;
  sosa:hosts <sensor/35-207306-844818-0/BMP282> ;
  sosa:hosts <actuator/35-207306-844818-0/tapticEngine> ;
  sosa:hosts <sensor/35-207306-844818-0/gps> .

<sensor/35-207306-844818-0/gps> a sosa:Sensor ;
  sosa:madeObservation <35-207306-844818-0/location/1> .

<35-207306-844818-0/location/1> a sosa:Observation ;
  sosa:observedProperty <location> ;
  sosa:resultTime
    "2017-08-18T00:00:12+00:00"^^xsd:dateTimeStamp ;
  sosa:hasResult [
   a geo:Point ;
   geo:lat "51.5"^^xsd:decimal ;
   geo:long "-0.12"^^xsd:decimal ;
   ] .

<actuator/35-207306-844818-0/tapticEngine> a sosa:Actuator ;
  sosa:actsOnProperty <tactileFeedback> ;
  sosa:usedProcedure <UIImpactFeedbackGeneratorAPI> .
```

Listing 2.4 shows how to model an observation involving a sample that is representative of the atmospheric pressure of Hurricane Maria at a given time. A `Sample` in SOSA is a `FeatureOfInterest` itself but also a `Result` of the act of `Sampling` as will be discussed in more detail below.

Listing 2.4: Sample modelling

```
<HurricaneMaria> a sosa:FeatureOfInterest ;
  rdfs:label "Hurricane Maria, 2017 season"@en .
<HurricaneMariaAPSampleAtStation1> a sosa:Sample;
  sosa:isSampleOf <HurricaneMaria>.

<Obs123> a sosa:Observation ;
  [...]
  sosa:featureOfInterest <HurricaneMariaAPSampleAtStation1> ;
  sosa:observedProperty <AtmosphericPressure> ;
  sosa:hasSimpleResult "101000 Pa"^^cdt:pressure ;
  sosa:resultTime "2017-09-19T23:00:00Z"^^xsd:dateTime .
```

The observation described above uses a custom datatype [21] (i.e. `cdt:pressure`) that leverages the Unified Code of Units of Measures, a code system intended to include *all* units of measures being contemporarily used in international science, engineering, and business. Such custom datatypes, although compatible with the RDF specification, are not yet recognized datatype IRIs and therefore not supported by current RDF and SPARQL engines. For Web applications we feel that such datatypes are useful and their support would allow an easy comparison of quantity values.

[3] See `https://www.w3.org/2003/01/geo/`

2.3. *Samples, Samplers, and Sampling*

A `Sample` is an object which is representative of a larger object or set of objects, created to support observations. Using a common core structure across different activities was one of the design goals of SOSA; hence, a `Sampler` is a device, a software or agent that is used by, or implements, a (sampling) procedure to create or transform one or more samples. The act by which a sampler creates a sample is called `Sampling`, and a sample is also a `Result`. Samplers can be hosted on platforms, e.g., together with sensors. While each sample is connected to some feature of interest using the `isSampleOf` relation, a sample may also be a feature of interest itself. Fig. 4 depicts the aforementioned classes and their relationships.

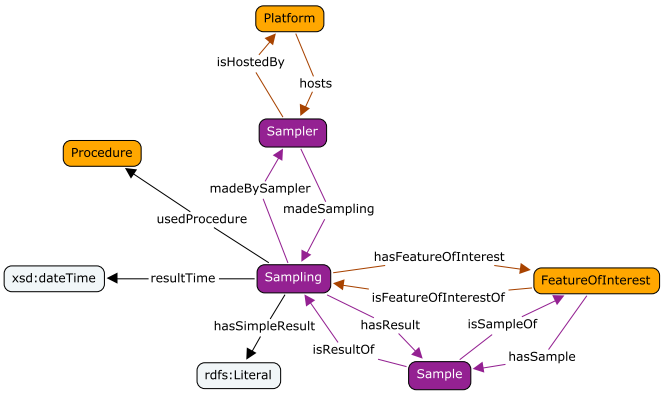

Figure 4: Overview of the SOSA Sampling perspective

Listing 2.5 below shows the result of applying a thermal drill to extract three samples to study the Antarctic ice sheet and then to take CO_2 observations from one of the samples. It also shows how to locate the sampling event in space and time. While one cannot determine the CO_2 level for the entire sheet, it is possible to use averages from the sampled values for approximation. Whether this approximation is meaningful for a specific study region and period or the entire sheet depends on our knowledge or theories about the development of Antartica's ice sheet and its uniformity.

Listing 2.5: Sampling modelling

```
<Antarctic_Ice_Sheet> a sosa:FeatureOfInterest ;
  sosa:hasSample <IceCore12>, <IceCore13>, <IceCore14> .

<IceCore12> a sosa:Sample ;
  sosa:isSampleOf <Antarctic_Ice_Sheet> ;
  sosa:isResultOf <WellDrilling4578> ;
  sosa:madeBySampler <ThermalDrill2> .

<WellDrilling/4578> a sosa:Sampling ;
 geo:lat -73.35 ;
 geo:long 9.32 ;
 sosa:hasResult <IceCore12> ;
 sosa:madeBySampler <ThermalDrill2> ;
 sosa:resultTime "2017-04-03T11:12:00Z"^^xsd:dateTime ;
 sosa:hasFeatureOfInterest <Antarctic_Ice_Sheet> .

<IceCore12Obs> a sosa:Observation ;
  sosa:hasFeatureOfInterest <IceCore12> ;
```

```
  sosa:observedProperty <CO2> ;
  sosa:hasSimpleResult 240 .
```

Additional classes and relationships to model relationships between samples are available via the *Sample Relations* vertical segmentation module; see section 5.2 of the SOSA/SSN spec.

2.4. Actuators and Actuations

SOSA also includes classes and relations to model the behaviour of actuation devices, called actuators, that carry out (actuation) procedures to change the state of the world. The modelling of actuations is analogous to the modelling of observations and sampling as it relies on the same core structure.

An `Actuation` is performed by an `Actuator` and yields a `Result`. An actuator is a device, software or agent that is used by, or implements, an (actuation) procedure that defines how changes of the state of the world are to be achieved. The actuator responds to an input, defined by the procedure and results in changes in the environment. The set of instructions for turning on and off an Internet of Things enabled light bulb is an example of a procedure. The activity of turning the light-bulb on/off is the actuation and the light bulb (or its socket) is the actuator. The difference with actuations, compared to observations, is that they may be used to model both, a record of how actuations have been performed (a log) and as how to interact with an actuation device (i.e., the procedure how to perform actuations) as well. The former is comparable to the use of the observation class, and the focus of SOSA, while the latter relies on additional axioms provided by the SSN extension to SOSA (i.e., the `System` concept and its properties `implements/implementedBy`) as well as on other ontologies and/or specifications that detail the functionality of an actuator further. For example, how much detail is provided to model inputs and outputs of the actuation procedure as well as the orchestration of multiple actuators is beyond the scope of both SOSA and SSN. Existing ontologies such as OWL-S [24] and execution frameworks such as WSMX [10] can be used together with lower-level specifications such as the W3C Thing Description[4] to model these details. With regards to SOSA, actuators are typically triggered by sensor outputs, i.e., observation results. The thing being acted on is the feature of interest of the actuation and the property being altered is an `ActuatableProperty`. To close the loop, such actuatable properties are typically also observable properties, e.g., the current state of the aforementioned light bulb. Put differently, the result of an actuation may be the stimulus of a new observation. Fig. 5 depicts the introduced classes and relations.

Listing 2.6 shows how `Actuation` of an Internet controlled light bulb has been performed.

Listing 2.6: Actuation modelling

```
<actuation/046677455286> a sosa:Actuation ;
  sosa:actuatableProperty <philips/046677455286/light> ;
  sosa:hasFeatureOfInterest <light> ;
  sosa:madeByActuator <actuator/philips/HJC42XB/bulb> ;
  sosa:hasResult [
```

[4] https://w3c.github.io/wot-thing-description/

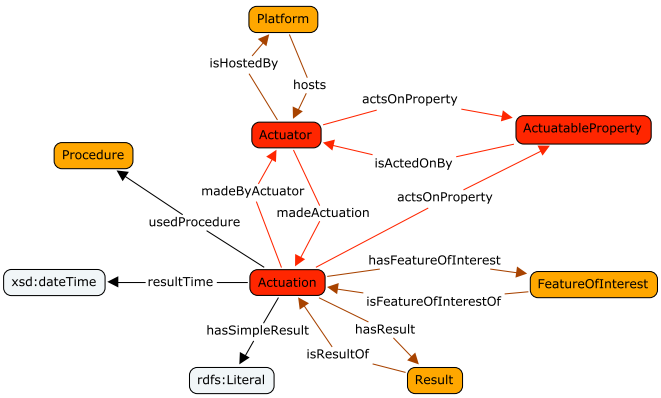

Figure 5: Overview of the SOSA Actuation perspective

```
  a qudt-1-1:QuantityValue ;
  qudt-1-1:numericValue "800"^^xsd:double ;
  qudt-1-1:unit qudt-unit-1-1:Lumen ] ;
 sosa:resultTime "2017-10-06T11:26:06Z"^^xsd:dateTime .
```

The actuation can be defined to have been made by the actuator (i.e., the light bulb) which also acts as a sensor, as it also allows to read out the current status (ON/OFF) of a light bulb and its current output as measured in lumen. Listing 2.7 shows how the light bulb `<actuator/philips/HJC42XB/bulb>` is hosted by the *Philips Hue Bridge* that acts as a controller for the actuator/sensor, and can be defined as a platform in SOSA.

Listing 2.7: Platform hosting modelling

```
<philips/46N7743619> a sosa:Platform ;
  rdfs:label "Philips Hue Bridge 46N7743619"@en ;
  rdfs:comment "Philips Hue Bridge - installed in living room"@en ;
  sosa:hosts <actuator/philips/HJC42XB/bulb> ;
  sosa:hosts <sensor/philips/HJC42XB/bulb> .

<actuator/philips/HJC42XB/bulb> a sosa:Actuator ;
  rdfs:label "Philips E27 Bulb - HJC42XB - Turn On/Off"@en ;
  sosa:actsOnProperty <philips/46N7743619/light> ;
  sosa:usedProcedure <philips/46N7743619/switchAPI>.

<sensor/philips/HJC42XB/bulb> a sosa:Sensor ;
  rdfs:label "Philips E27 Bulb - HJC42XB - Read Lumen"@en ;
  sosa:observes <philips/46N7743619/light> .
```

3. Vertical Segmentation Alignments

Vertically segmented modules that import SOSA add higher levels of ontological commitment on top of SOSA by defining new axioms. They are provided for users to either migrate from the old SSN to the new version or to interlink or map data expressed according to standards such as O&M and OBOE to SSN. SOSA as the core is independent of higher level modules and does not import any other ontologies and its axiomatization is deliberately limited as discussed above. We do not expect users of SOSA to use any of the provided alignment modules (as described in Section 6 of the standard) as they would require the import of an ontology file (which is not common practice in the schema.org community [9]). It is worth noting here, though, that SOSA was developed with these ontologies

in mind to act as a common interoperability fallback level between them. An example of that alignment was already given in Listing 4.2 where we used SOSA in combination with PROV-O.

To give a few more examples of how SOSA relates to other ontologies, `sosa:Observation` is equivalent to `o&m:Observation`, and is a superclass of `oboe:Measurement`. It should not be confused with `oboe:Observation`, which is a *collection* of measurements of different properties of the same feature of interest. All of these are, however, subclasses of `dul:Event` as well as `prov:Activity`. While a `sosa:Procedure` can be aligned directly to an `oboe:Protocol` and as a subclass of `prov:Plan`, the alignment to an `o&m:Process` is defined by an equivalent class relationship to the *Union* of `sosa:Sensor` and `sosa-om:ObservationProcedure`, where the latter is a subclass of `sosa:Procedure` in which every `usedProcedure`$^-$ points to an `sosa:Observation`. More formally:

$$\begin{aligned} \texttt{ObservationProcedure} \sqsubseteq\ & \texttt{sosa:Procedure} \sqcap \\ & \forall \texttt{sosa:usedProcedure}^-.\texttt{sosa:Observation} \sqcap \\ & \exists \texttt{sosa:usedProcedure}^-.\texttt{sosa:Observation} \\ \texttt{o\&m:Process} \equiv\ & \texttt{sosa:Sensor} \sqcup \\ & \texttt{ObservationProcedure} \end{aligned}$$

4. Using SOSA on the Web

SOSA was engineered to provide a lightweight core vocabulary to model observations, actuations and samplings and is aimed at a much broader audience than the original SSN, including Web developers who are accustomed to JSON serializations and at most schema.org style annotations. Considering that already 38% of all URIs crawled in the Web Data Commons project [30] use some form of RDF for metadata modelling, it is expected that practitioners dealing with sensing and actuating devices on the Web will increasingly use metadata to specify the capabilities of these devices and the way that observations and actuations on these devices have been performed.

Schema.org is the de-facto standard vocabulary [9] that is embraced by Web developers and helps to integrate data across applications and data formats. Schema.org descriptions can be written using markup attributes in HTML (i.e., using RDFa, Microdata, or JSON-LD as serialisation formats). As these serialisations provide value for developers and publishers, we will discuss briefly how to use SOSA with Schema.org.

4.1. SOSA + Schema.org RDFa

RDFa 1.1 provides a set of attribute-level extensions to embed RDF in HTML5 and XHTML5. Revisiting our iPhone example from above, another observation, made a minute later by the same phone, could be modelled using RDFa as described in Listing 4.1.

Listing 4.1: RDFa 1.1. serialization

```
<div typeof="sosa:Observation" about="ex:data/observation/346345">
 <div rel="rdf:type" resource="schema:Action"></div>
 <div rel="sosa:hasFeatureOfInterest">
  <div typeof="sosa:FeatureOfInterest" about="ex:earthAtmosphere">
   <div property="rdfs:label" xml:lang="en" content="Earth
        Atmosphere"></div>
 </div> </div>
 <div rel="sosa:observedProperty" resource="ex:BMP282/AP"></div>
 <div rel="sosa:madeBySensor">
  <div typeof="sosa:Sensor"
       about="ex:sensor/35-207306-844818-0/BMP282">
   <div rel="sosa:observes" resource="ex:BMP282/AP"></div>
   <div property="rdfs:label" xml:lang="en" content="Bosch
        Sensortec BMP282"></div>
 </div> </div>
 <div property="sosa:resultTime" datatype="xsd:dateTime"
      content="2017-06-06T12:37:12+00:00"></div>
 <div property="sosa:hasSimpleResult" datatype="cdt:ucum"
      content="1022.05 hPa"></div>
 <div property="schema:location" typeof="Place">
  <div property="address" typeof="schema:PostalAddress">
   <div property="schema:addressLocality" xml:lang="en"
        content="Canberra"></div>
   <div property="schema:addressCountry" xml:lang="en"
        content="AU"></div>
 </div> </div> </div>
```

The example also integrates the use of Schema.org to further define the location of an observation. Defining a SOSA observation to be also of type schema.org `Action` allows the ontology to use the location property for the action to specify the place where the observation took place, e.g., above in Canberra, Australia. A Schema.org action has several other properties that can be used to provide further detail about the observation, including, for example, participants in the action beyond the one who performed it (modelled through the schema:agent relation, which is equivalent to `sosa:madeBySensor`).

4.2. SOSA + PROV-O JSON-LD

JSON-LD is a JSON-based format to serialize Linked Data. JSON-LD was a reaction to the popularity of JSON, a lightweight, language-independent data interchange format for the Web. It has become the language of choice for the majority of web developers as it is easy to parse and easy to generate. In Listing 4.2 we show how to serialize a SOSA example in JSON-LD and how to model an agent and what role he played in the act of sensing, using PROV-O.

Listing 4.2: JSON-LD serialization

```
{
 "@graph": [
{
  "@id": "ex:distancemeter/838725",
  "@type": [ "sosa:Sensor" ],
  "rdfs:label": { "@value": "Leica Disto D2 - 838725"}
},{
  "@id": "ex:observation/1087",
  "@type": [ "prov:Activity", "sosa:Observation" ],
  "prov:qualifiedAssociation": { "@id": "_:N5bc2a9" },
  "sosa:hasResult": { "@id": "_:Nc90f75" },
  "sosa:madeBySensor": { "@id": "ex:distancemeter/838725"
  },
  "sosa:observedProperty": { "@id": "ex:section/316/length" }
},{
  "@id": "_:N5bc2a9",
  "@type": "prov:Association",
  "prov:agent": { "@id": "ex:bobthebuilder" },
  "prov:hadRole": { "@id": "ex:structuralEngineer1" },
}]}
```

The example shows an observation of the length of a stretch of road (i.e. `ex:section/316/length`) that has been made using the Leica Disto D2 laser distance meter. Since it may be important in this case (for legal/contractual reasons), to record who used

the instrument and in which role, the PROV-O ontology can be used to state that "Bob the Builder" made `ex:observation/1087` in his role as a structural engineer. Since SOSA has been modelled in an event-centric way, an observation maps to an activity in PROV-O (cf. Section 6.5 in the SSN spec). To associate an agent (`ex:bobthebuilder`) and the role (`ex:structuralEngineer1`) he played in the activity, a qualified association was used in the example above that uses a blank node (`_:N5bc2a9`) for associating the two properties. However, this blank node could also be identified by an IRI instead, e.g. `bobAsStructuralEngineer1`, if further statements should be made about that association. Other statements can also be made about the `prov:Activity`, including when and if another observation was made in the past, who authorized it, and so on.

5. Modelling with SOSA

While we have already provided numerous examples and guidance on modelling with SOSA, this section discusses some common cases that may arise in practice.

5.1. Collections

There are many reasons to group a set of observations, actuations, or sampling activities together. Avoiding redundancy and reducing storage size is one of them. Given that a sensor may generate thousands of observations during a campaign, generating an assertion for each of these observations to connect them to the used sensor via `madeBySensor` and to the used procedure via `usedProcedure` becomes bothersome. While SOSA and SSN do not offer a specific collections class to group observations, actuations, or sampling activities, terminological axioms may take over this role. For example, in case of observations all being taken by a specific sensor, say `Sensor1`, one would specify a subclass of `Observation` as follows.

$$\texttt{S1Obs} \sqsubseteq \texttt{Observation} \sqcap \exists \texttt{madeBySensor}.\{\texttt{Sensor1}\}.$$

Instead of connecting every observation with the same sensor, one would simply define observations to be of type S1Obs as shown in Listing 5.1.

Listing 5.1: Collections of Observations

```
<S1Obs1> a sosa:S1Obs ;
  rdfs:label "Observation1 being a Sensor1Observation (S1Obs),
       i.e., being taken by Sensor1."@en ;
  [...]
<S1Obs2> a sosa:S1Obs ;
  [...]
```

The same applies to procedures, features of interest, observed properties, and so forth, as well as any combination thereof. Classes related to deployments are offered by the SSN module and can be used to group observations by campaigns.

Finally, another question that may arise in the context of storing observations relates to data cataloging and serving. SOSA does not provide classes for information objects, datasets, and catalogs as they are provided by W3C recommendations such as the Data Catalog Vocabulary (DCAT) [22]. Similarily, storage, service, and streaming can be handled using methods and tools developed during the intitial SSN-XG work [12, 19, 15].

5.2. Individuals versus Classes

Another common issue is the decision between using individuals or classes, e.g., for observed and actuatable properties as well as procedures, to model the interaction with controlled vocabularies. SOSA does not recommend a specific strategy but relies on either SKOS, language elements such as OWL2's *punning*, or modeling patterns such as casting between individuals (e.g., enumerated literals from code lists) and classes as shown below; see [18] for details.

$$\text{ClassName} \sqsubseteq \exists \text{hasType}.\{\text{classname}\}$$
$$\exists \text{hasType}.\{\text{classname}\} \sqsubseteq \text{ClassName}$$

Returning to the nitrate concentration example given above, a terminological axiom such as

$$\texttt{cv:NitrateConcentration} \equiv \exists \texttt{sosa:observedProperty}.\{\texttt{cv:NO3\_concentration}\}.$$

would be used and also enable the creation of hierarchies, e.g., to say that the *NitrateConcentration* observed property (taken from a code list) is a sub class of *Concentration*.[5]

5.3. Events versus Records

As alluded to above, perhaps the key conceptual revision compared to the SSO pattern is that observations are now conceived as acts or events. This enabled the establishment of the core structure, with a common pattern for observation, actuation and sampling, which is consistent with other observation models in ontologies, and also makes alignment with PROV-O easy. However, it represents a break with the original SSN ontology that built upon the SSO pattern in which an observation was effectively a record or description of an observation context (a kind of `dul:Situation`), rather than an activity in the world (a `dul:Event`) (see [6] for more discussion). The use of a new namespace for the core classes, as part of SOSA, helps avoid confusion with implementations of the original SSN, but care must be taken if reasoning and interpreting over graphs that includes data represented using both ontologies. In the standardization process we have provided alignment files between the old SSN and SOSA/SSN. A detailed description on how terms in SOSA relate to the old SSN is outside the scope of this paper and we refer the interested reader to the specification [11].

6. Evidence of use of SOSA

The Spatial Data on the Web Working Group collected 51 use cases, from which 62 requirements were derived, which informed the modelling of SOSA and SSN.[6] Based on these requirements, the SOSA ontology was designed

[5] See here for an SKOS-based example `http://environment.data.gov.au/def/property/nitrate_concentration`.

[6] The use cases and requirements can be accessed at `https://www.w3.org/TR/sdw-ucr/`.

to reuse the concepts and properties that have been previously defined in the original SSN ontology, along with additional features specified and requested in previous work such as [5] and [6]. Of the 13 concepts and 21 properties in SOSA, three concepts (i.e. `Sensor`, `Platform` and `FeatureOfInterest`) and eleven properties (i.e. `hosts`, `isHostedBy`, `usedProcedure`, `hasFeatureOfInterest`, `madeObservation`, `madeBySensor`, `observedProperty`, `hasResult`, `isResultOf`, `phenomenonTime`) are equivalent to a similar term in the original SSN. New or changed concepts relate to the core structure (`Procedure`, `Result`, `ObservableProperty`, `ActuatableProperty`), to the expansion of scope to cover sampling and actuation (`Sampler`, `Sample`, `Sampling`, `Actuator`, `Actuation`), and to the modified interpretation for `Observation`.

The many implementations of the original SSN provide evidence for the utility of those elements carried over from the earlier work, and several implementations using the new elements of SOSA have already been established, including the modelling of samples in Geoscience Australia[7], the modelling of oceanographic time series in the South Adriatic Pit (Eastern Mediterranean) by Center for Marine Environmental Sciences, University of Bremen[8], and a dataset of measurements of a meteorological station by Irstea[9].

A complete list of the 23 ontologies that already (re)use SOSA and the 23 datasets that use SOSA classes and properties to define data in their applications can be found in the SSN Usage document [8] that records implementation evidence in the SDW as a required part of the W3C standardization process.

7. Conclusion

With 8.4bn connected things in use worldwide in 2017 and with more than 31% of websites [9] using Schema.org annotations, there is a strong desire by the Web community for a lightweight vocabulary to describe sensors, actuators, and samplers and the acts they can perform. In this article we presented SOSA, a lightweight ontology that represents the core of the new Semantic Sensor Network ontology, a recommendation built in a joint effort by the W3C and OGC, that was specifically designed with the requirements of Web developers, domain scientists, and Linked Data engineers in mind. With its event-centric view, SOSA aligns well with other standards (i.e., PROV-O, O&M) and with its Schema.org style *domainIncludes* and *rangeIncludes* annotation properties to provide an informal semantics, the ontology can easily be used in existing applications that support Schema.org annotations to describe IoT devices and their capabilities. The newly established *Spatial Data on the Web Interest Group* will promote the adoption of SOSA will work on a proper layering (binding) of the newly developed W3C Thing description to SOSA.

[7] `http://pid.geoscience.gov.au/sample/`

[8] `https://markusstocker.github.io/eyp-fixo3-ld/browser`

[9] `http://ontology.irstea.fr/pmwiki.php/Site/Weather2017`

Acknowledgements

The SOSA ontology was developed as part of the OGC/W3C Spatial Data on the Web Working Group, see `https://www.w3.org/2000/09/dbwg/details?group=75471&public=1` for the list of members. The efforts of W3C staff Phil Archer and François Daoust were invaluable in enabling the successful completion of the work through to publication as a W3C Recommendation and OGC Standard. The authors acknowledge partial support from NSF (award number 1540849).

8. References

[1] ISO 19156:2011 Geographic information - Observations and measurements. International Standard, December 2011. `https://www.iso.org/standard/32574.html`.

[2] Mike Botts and Alexandre Robin. OpenGIS sensor model language (SensorML) implementation specification. OpenGIS Implementation Specification OGC 000, 2007.

[3] Michael Compton, Payam Barnaghi, Luis Bermudez, Raul Garcia-Castro, Oscar Corcho, Simon Cox, John Graybeal, Manfred Hauswirth, Cory Henson, Arthur Herzog, Vincent Huang, Krzysztof Janowicz, W. David Kelsey, Danh Le Phuoc, Laurent Lefort, Myriam Leggieri, Holger Neuhaus, Andriy Nikolov, Kevin Page, Alexandre Passant, Amit Sheth, and Kerry Taylor. The ssn ontology of the w3c semantic sensor network incubator group. *Web Semantics: Science, Services and Agents on the World Wide Web*, 17(0):25–32, 2012.

[4] Michael Compton, Cory Henson, Laurent Lefort, Holger Neuhaus, and Amit Sheth. A survey of the semantic specification of sensors. In *Proc. of 2nd Conference on the SSN-Volume 522*, pages 17–32. CEUR, 2009.

[5] Simon J D Cox. Observations and measurements. *OGC Best Practices Document. OGC*, page 21, 2006.

[6] Simon J D Cox. Ontology for observations and sampling features, with alignments to existing models. *Semantic Web*, 8(3):453–470, 2017.

[7] Aldo Gangemi and Valentina Presutti. Ontology design patterns. In *Handbook on ontologies*, pages 221–243. Springer, 2009.

[8] Raúl García-Castro, Armin Haller, and Nandana Mihindukulasooriya. On the usage of the SSN ontology. W3C Note, November 2017. `https://w3c.github.io/sdw/ssn-usage/`.

[9] R.V. Guha, Dan Brickley, and Steve Macbeth. Schema.org: Evolution of structured data on the web. *ACM Queue*, 13(9), 2015.

[10] Armin Haller, Emilia Cimpian, Adrian Mocan, Eyal Oren, and Christoph Bussler. Wsmx-a semantic service-oriented architecture. In *Web Services, 2005. ICWS 2005. Proceedings*, pages 321–328. IEEE, 2005.

[11] Armin Haller, Krzysztof Janowicz, Simon J D Cox, Danh Le Phuoc, Kerry Taylor, Maxime Lefrançois, Rob Atkinson, Raúl García-Castro, Joshua Lieberman, and Claus Stadler. Semantic Sensor Network Ontology. W3C Recommendation, October 19 2017. `https://www.w3.org/TR/vocab-ssn/`.

[12] Cory A Henson, Josh K Pschorr, Amit P Sheth, and Krishnaprasad Thirunarayan. Semsos: Semantic sensor observation service. In *Collaborative Technologies and Systems, 2009. CTS'09. Int. Symposium on*, pages 44–53. IEEE, 2009.

[13] Matthew Horridge, Samantha Bail, Bijan Parsia, and Ulrike Sattler. The cognitive complexity of owl justifications. *The Semantic Web–ISWC 2011*, pages 241–256, 2011.

[14] Krzysztof Janowicz. Observation-driven geo-ontology engineering. *Transactions in GIS*, 16(3):351–374, 2012.

[15] Krzysztof Janowicz, Arne Bröring, Christoph Stasch, Sven Schade, Thomas Everding, and Alejandro Llaves. A restful proxy and data model for linked sensor data. *Int. Journal of Digital Earth*, 6(3):233–254, 2013.

[16] Krzysztof Janowicz and Michael Compton. The stimulus-sensor-observation ontology design pattern and its integration into the semantic sensor network ontology. In *Proc. of the 3rd Conference on the SSN-Volume 668*, pages 64–78. CEUR, 2010.

[17] Sebastian Kaebisch and Takuki Kamiya. Web of Things (WoT) Thing Description. W3C Working Draft 5 April 2018, April 2018. `https://www.w3.org/TR/2018/WD-wot-thing-description-20180405/`.

[18] Adila A Krisnadhi, Pascal Hitzler, and Krzysztof Janowicz. On the capabilities and limitations of owl regarding typecasting and ontology design pattern views. In *International Experiences and Directions Workshop on OWL*, pages 105–116. Springer, 2015.

[19] Danh Le-Phuoc, Minh Dao-Tran, Josiane Xavier Parreira, and Manfred Hauswirth. A native and adaptive approach for unified processing of linked streams and linked data. In *Int. Semantic Web Conference*, pages 370–388. Springer, 2011.

[20] Tim Lebo, Satya Sahoo, and Deborah McGuiness. PROV-O: The PROV Ontology. W3C Recommendation, April 30 2013. `https://www.w3.org/TR/prov-o/`.

[21] Maxime Lefrançois and Antoine Zimmermann. Custom Datatypes - Towards a web of Linked Datatypes. Technical Report, March 2018. `https://ci.mines-stetienne.fr/lindt/v2/custom_datatypes.html`.

[22] Fadi Maali, John Erickson, and Phil Archer. Data catalog vocabulary (dcat). *W3C Recommendation*, 16, 2014.

[23] Joshua Madin, Shawn Bowers, Mark Schildhauer, Sergeui Krivov, Deana Pennington, and Ferdinando Villa. An ontology for describing and synthesizing ecological observation data. *Ecological informatics*, 2(3):279–296, 2007.

[24] David Martin, Mark Burstein, Jerry Hobbs, Ora Lassila, Drew McDermott, Sheila McIlraith, Srini Narayanan, Massimo Paolucci, Bijan Parsia, Terry Payne, et al. Owl-s: Semantic markup for web services. *W3C member submission*, 22:2007–04, 2004.

[25] Daniel Mekonnen, David Price, Jack Hodges, James E. Masters, Simon Cox, and Steve Ray. Quantities, units, dimensions and types (qudt) schema - version 2.0. Technical report, 2017. `http://qudt.org`.

[26] Matthew Perry and John Herring. Ogc geosparql-a geographic query language for rdf data. *OGC Implementation Standard. Sept*, 2012.

[27] Alan Rector, Nick Drummond, Matthew Horridge, Jeremy Rogers, Holger Knublauch, Robert Stevens, Hai Wang, and Chris Wroe. Owl pizzas: Practical experience of teaching owl-dl: Common errors & common patterns. In *Int. Conference on Knowledge Engineering and Knowledge Management*, pages 63–81. Springer, 2004.

[28] Nick Russell, Arthur Ter, Wil M. P. Aalst, and Nataliya Mulyar. Workflow Control-Flow Patterns: A Revised View. *Technical Report BPM-06-22*, 01 2006.

[29] Ashok Malhotra Steve Speicher, John Arwe. Linked Data Platform 1.0. W3C Recommendation, February 2015.

[30] Web Data Commons. RDFa, Microdata, Embedded JSON-LD, and Microformats Data Sets. Technical report, November 2017. `http://webdatacommons.org/structureddata/2017-12/stats/stats.html`.