The Modular SSN Ontology: A Joint W3C and OGC Standard Specifying the Semantics of Sensors, Observations, Sampling, and Actuation
Armin Haller, Krzysztof Janowicz, Simon Cox, Maxime Lefrançois, Kerry Taylor, Danh Le Phuoc, Joshua Lieberman, Raúl García-Castro, Rob Atkinson, Claus Stadler

To cite this version:

HAL Id: hal-01885335
https://hal.archives-ouvertes.fr/hal-01885335
Submitted on 1 Oct 2018

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L’archive ouverte pluridisciplinaire HAL, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d’enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.
The Modular SSN Ontology: A Joint W3C and OGC StandardSpecifying the Semantics of Sensors, Observations, Sampling, and Actuation

Armin Haller\textsuperscript{a,∗}, Krzysztof Janowicz\textsuperscript{b}, Simon J D Cox\textsuperscript{c}, Maxime Lefrançois\textsuperscript{d}, Kerry Taylor\textsuperscript{a}, Danh Le Phuoc\textsuperscript{e}, Joshua Lieberman\textsuperscript{f}, Raúl García-Castro\textsuperscript{g}, Rob Atkinson\textsuperscript{h}, and Claus Stadler\textsuperscript{i}

\textsuperscript{a} Research School of Computer Science, Australian National University, Canberra, Australia
\textsuperscript{b} Geography Department, University of California, Santa Barbara, CA, USA
\textsuperscript{c} Land and Water, CSIRO, Melbourne, Australia
\textsuperscript{d} Univ Lyon, MINES Saint-Étienne, CNRS, Laboratoire Hubert Curien UMR 5516, Saint-Étienne, France
\textsuperscript{e} Open Distributed Systems, Technische Universität Berlin, Germany
\textsuperscript{f} Center for Geographic Analysis, Harvard University, Boston, MA, USA
\textsuperscript{g} Ontology Engineering Group, Universidad Politécnica de Madrid, Madrid, Spain
\textsuperscript{h} Metalinkage, Wollongong, Australia
\textsuperscript{i} Institut für Informatik, Universität Leipzig, Leipzig, Germany

E-mails: armin.haller@anu.edu.au, kerry.taylor@anu.edu.au, jano@geog.ucsb.edu, simon.cox@csiro.au, maxime.lefrancois@emse.fr, danh.lephuoc@tu-berlin.de, flieberman@fas.harvard.edu, rgarcia@fi.upm.es, rob@metalinkage.com.au, cstadler@informatik.uni-leipzig.de

Editor: Pascal Hitzler, Wright State University, USA

Solicited reviews: Eva Blomqvist, Linköping University, Sweden; Alessandro Oltramari, Bosch Research and Technology Center, USA; Freddy Lucie, Accenture Tech Labs Ireland, and INRIA, France

Abstract. The joint W3C (World Wide Web Consortium) and OGC (Open Geospatial Consortium) Spatial Data on the Web (SDW) Working Group developed a set of ontologies to describe sensors, actuators, samplers as well as their observations, actuation, and sampling activities. The ontologies have been published both as a W3C recommendation and as an OGC implementation standard. The set includes a lightweight core module called SOSA (Sensor, Observation, Sampler, and Actuator) available at: http://www.w3.org/ns/sosa/, and a more expressive extension module called SSN (Semantic Sensor Network) available at: http://www.w3.org/ns/ssn/. Together they describe systems of sensors and actuators, observations, the used procedures, the subjects and their properties being observed or acted upon, samples and the process of sampling, and so forth. The set of ontologies adopts a modular architecture with SOSA as a self-contained core that is extended by SSN and other modules to add expressivity and breadth. The SOSA/SSN ontologies are able to support a wide range of applications and use cases, including satellite imagery, large-scale scientific monitoring, industrial and household infrastructures, social sensing, citizen science, observation-driven ontology engineering, and the Internet of Things. In this paper we give an overview of the ontologies and discuss the rationale behind key design decisions, reporting on the differences between the new SSN ontology presented here and its predecessor [9] developed by the W3C Semantic Sensor Network Incubator group (the SSN-XG). We present usage examples and describe alignment modules that foster interoperability with other ontologies.

Keywords: Ontology, Sensor, Actuator, Observation, Actuation, Sampling, Linked Data, Web of Things, Internet of Things
1. Introduction

Sensors are a major source of data available on the Web today. The trend towards making cities, offices, and homes ‘smarter’ by turning them into sensor-rich environments [31] drives the demand for specifications that describe how to model and publish sensor and actuator data as well as how to foster interoperability across platforms on the Web or other data infrastructures.

Sensor readings are often provided only as raw numeric values, but any searching, reusing, integrating, or interpreting of these data requires more than just the observation results. Of equal importance for the proper interpretation of these values is contextual information about the studied feature of interest, such as a river, the observed property, such as flow velocity, the utilized sampling strategy, such as the specific locations or sampling stations and times at which the velocity was measured, the procedures followed to produce a result, and a variety of other information.

The Open Geospatial Consortium’s (OGC) Sensor Web Enablement (SWE) standards [6, 26, 49] provide a framework for describing sensors and observations, as well as encodings for their data and service interfaces for interacting with them. SWE implementations followed the style of other OGC standards, relying on web-hosted service calls and XML payloads, which have limited compatibility with the more recent and widely used web platforms.

A W3C Incubator Group was created in 2009 with the goal, among others, of defining an OWL ontology that would, to some degree, reflect the SWE standards. That group produced the original Semantic Sensor Network ontology (SSNX) [9, 36]. SSNX was designed to be consistent with W3C endorsed best practices on publishing data on the Web, which recommend a Linked Data approach, allowing sensor data, for example, to be published as part of a global and densely interconnected graph of data [42]. The SSNX ontology has been the basis for many subsequent research initiatives and some operational deployments. However, users of the original ontology have identified a number of potential points of improvement, inconsistencies with related vocabularies such as the Ontology of units of Measure (O&M) [12, 49], as well as extensions to cover new aspects of sensing that have become more relevant, such as actuation, a key element of the Internet of Things (IoT).

Four years after the publication of the first version, work started on an update and formal standardization of the SSNX ontology, this time jointly led by the W3C and the OGC, to address the feedback gathered on the usage of the ontology and lessons learned. This activity resulted in the publication of the new Semantic Sensor Network (SSN) modular ontology, designed to provide a flexible but coherent perspective for representing the entities, relations, and activities involved in sensing, sampling, and actuation [22]. The main innovation of this formal standard version of SSN is the separation of SSN into ontology modules, the central of which is the Sensor, Observation, Sample, and Actuator (SOSA) ontology, providing a lightweight core for SSN. SOSA aims to broaden the target audience and application areas that can make use of Semantic Web ontologies. Other SSN modules add additional terms, introduce additional ontological commitments, and/or clarify and support the alignment of SSN with other ontologies.

This paper does not aim at providing an exhaustive description of the SSN ontology, since that can be found in the specification [22], but to motivate the development decisions and the design patterns followed while indicating the most substantial changes made since the initial release of the ontology. Throughout this paper we will introduce concepts from SSN (and SOSA) in detail, supported by examples of its use in a smart home case study. As will be detailed below, SOSA [30] is a self-contained pattern designed for reuse and to meet the demands of a larger, schema.org-style target audience. Hence, SOSA is only introduced here to a degree necessary to understand its relation to SSN and because SSN imports and extends SOSA classes and relations.

The remainder of the paper is structured as follows. Sec. 2 briefly reviews the origins of the SSN ontology. Sec. 3 explains the rationale behind the main changes to the core of SSN, with observations modeled as events alongside the related sampling and actuation activities which are introduced as well. The overall modular structure of the SSN ontology is described in Sec. 4. A general description of the new, simplified SSN ontology is provided in Sec. 5, along with examples and explanations for the changes from SSNX. Sec. 6 and 7 describe the horizontal and vertical ontology modules that SSN imports or that are imported by SSN, respectively. We provide an overview of what

---

*Corresponding author. E-mail: armin.haller@anu.edu.au.

1In this paper SSNX will denote the 2011 version of the ontology produced by the W3C Incubator, with SSN used for the new version standardized through W3C/OGC and described here.
SSN is not intended to model in Sec. 8, and report in Sec. 9 on implementation evidence for SSN, which was crucial for it to become an OGC/W3C standard.

2. Original SSN and Antecedents

The development of SSN and SOSA has been strongly influenced by the SSNX ontology [36] and models from the OGC’s Sensor Web Enablement initiative (SWE) [5].

Starting in 2002 SWE developed a generic framework for delivering sensor data. The Sensor Observation Service defines a HTTP query interface for sensor and observation data [8], following the pattern successfully established by OGC starting with the Web Map Service [16]. Returned data conforms with the Sensor Model Language (SensorML) [6] and with the XML implementation of O&M [12, 49]. SensorML and O&M provide complementary viewpoints. SensorML is ‘provider-centric’ and encodes details of the sensor along with a stream of (typically) raw observation data. In contrast, O&M was designed to be more ‘user-centric’, with the target of the observation and the observed property as first-class objects.

A key aspect of the O&M model is that it separately defined classes for the description of the observation event, the real world feature of interest being observed, and the platform, procedure and/or system responsible for the observation. Geometries or other spatial location properties can be attached separately to features of interest, observations, platforms, and sensors, so the model can accommodate remote sensing and ex-situ observation scenarios alongside in-situ monitoring [32, §5.16][13, 26]. O&M also includes a model for sampling, since most practical observations are made on a subset of, or proxy for, the ultimate feature of interest.

O&M is only one of several similar conceptual models for observations and their results. A selection of these, also including OBOE [43], Sensei [2] and Seronto [55], were reviewed by the W3C Semantic Sensor Network Incubator Group, and guided the development of the SSNX ontology [36].

The SSNx ontology was ultimately built around a fundamental conceptual model implemented as an ontology design pattern called the Stimulus Sensor Observation (SSO) pattern [29] that was aligned to the Dolce-Ultralite upper ontology (DUL) [45]. SSO was intended to provide a minimal common ground for ontologies for the use on the Semantic Sensor Web.

Drawing on considerable implementation and application experience with SSNX, as well as with sensor and observation models and ontologies more broadly, the new SSN ontology presented here is set out to address changes in scope and audience, new technical developments and shortcomings of the initial work.

One such significant change in the extent of the audience of SSN has been the emergence of lightweight vocabularies such as schema.org and their increasing use on the Web [20] since SSNX had been published. It led to a requirement to provide a lightweight ontology with minimal ontological commitment [32, §5.22], the SOSA core, that can be used by Web developers to annotate their IoT APIs using simple serialization formats such as JSON-LD, Microdata or RDFa 1.1 without the need to import any other ontology, including upper level ontologies (as required by SSNX).

New technical developments included particularly the emergence of complex actuation devices on the Web of Things, such as home automation devices like Google Nest, personal assistants like Amazon Echo, Apple Siri or Microsoft Cortana and personal camera drones. To be able to model the smart instrumentation of these devices and the environments they operate in more generally, the actuator and actuation perspective has been added to SSN.

The sections below highlights these significant changes and some of the other changes in the new SOSA/SSN ontology.

3. Observation, Sampling and Actuation events

Following the working group’s Use Cases and Requirements analysis [32], the scope of the revised ontology has first been reduced by removing the concepts for stimulus, systems, measurement and system capabilities from the core (either for reasons of lack of implementation evidence or in response to a requirement [32, §5.47]), then expanded beyond sensors and their observations by including classes and properties for the closely linked concepts of actuation [32, §5.16] and sampling [32, §5.16].

Figure 1 provides an overview of the classes and properties in the core of the SSN ontology, showing how the three applications use the same pattern.

3.1. Sensors and Observations

The core of the SSNX ontology placed the sensor stimulus as the critical ‘event’ in the observation pro-
Fig. 1. Overview of the core structure of the SSN ontology, emphasizing the common patterns used by the three activities with classes stacked where they play a similar role. The elements shown are from both SOSA and SSN modules, with classes and types from external vocabularies indicated with a namespace prefix and uncolored. Classes and properties in blue relate to observations, red to sampling, and green to actuation, and classes in yellow are used across all three applications. A full set of inverse properties are defined in the ontology, but only a subset are shown in this figure.

success. Observations have therefore been modelled as a social construct (oldssn:Observation ⊑ dul:Situation), i.e. observations were contexts for interpreting incoming stimuli and fixing parameters such as time and location. While this is sound conceptual modeling, in practice the stimulus class was rarely instantiated. This is because a stimulus triggers the sensor to perform an observation, and, thus, resides outside the scope of typical sensor and observation applications and use cases. In subsequent work focusing on the alignment of SSN with the W3C Provenance ontology [34] the absence of a class representing an observation as an ‘act of sensing’ was noted [10, 14]. The revised model focuses on the complete observation as an event (sosa:Observation ⊑ dul:Event), completed when the result is available, i.e. acts of carrying out an (observation) procedure in order to estimate or calculate a value of a sosa:ObservableProperty of a sosa:FeatureOfInterest or a sosa:Sample thereof. This modelling is aligned with other standard ontologies such as the OGC Observations and Measurements [13, 26] (i.e. sosa:Observation ≡ so19156-om:OM_Observation) and the PROV Ontology (i.e. sosa:Observation ⊑ prov:Activity) where entities (i.e. sosa:Sensors in SSN) perform activities (i.e. sosa:Observations in SSN). The new model is also in line with the view of most practitioners, e.g., the event-oriented O&M pattern used in INSPIRE implementations². Furthermore, it provides a pattern and terminology that can be used analogically for sampling and actuation activities.

In the process of separating SSN from DUL, also the semantics of the sosa:Sensor class has been reconsidered. SSN has been criticized for its partially inconsistent handling of virtual sensors (including software and simulations) and related classes and properties. In particular, the comment in the oldssn:Sensor class suggested that a sensor can also be “computational methods, a laboratory setup with a person following a method, or any other thing that can follow a Sensing Method to observe a Property”. However, the oldssn:Sensor class was defined as a subclass of dul:PhysicalObject, making the comment inconsistent with the class definition. Sosa/SSN addresses this issue (and requirement §5.19 [32]) by allowing all major classes to be virtual, allowing humans and other animals as agents that perform observations, actuation, or sampling activities. In the optional alignment to DUL, sosa:Sensors, but also sosa:Platforms, and ssn:Systems are now defined as a subclass of the higher-level dul:Object class rather than the more specific dul:PhysicalObject class.

²https://inspire.ec.europa.eu/id/document/tg/d2.9-o%26m-swe
The SOSA pattern models `sosa:Sensors` as entities that make `sosa:Observations` about some `sosa:ObservableProperty` of a `sosa:FeatureOfInterest`. From the viewpoint of foundational ontologies such as DOLCE [18], endurants participate in perdurants. Consequently, the sensors, features of interest, samples, and so forth participate in the observation event in different roles. For instance, a sensor (typically) transforms a stimulus into a result thereby also setting the temporal bounds (start and end) of an observation. The feature of interest is the bearer of the property under consideration, and so on. To improve readability and to stay in line with the literature about sensors and observations we will say that a sensor triggered, initiated, performed, or took, an observation about a property of some feature instead of speaking about endurants and their participation in perdurants. Finally, it is also important to differentiate between the conceptual model of an observation as an event and the data models and data structures used to represent an observation as a record in a database, e.g., as served by a semantically-enabled Sensor Observation Service (SOS) [24, 28].

Throughout the paper we will use an exemplary smart house that has been modelled using the SSN ontology. Fig. 2 gives an overview of the SOSA/SSN instances that we will refer to in the listings throughout the paper.

Listing 3.1 uses SOSA/SSN to describe an observation of the electric consumption in the kitchen of our exemplary smart house #134 made by sensor #927.

### Listing 3.1: Observation Modelling

```
@prefix cdt: <http://www.w3id.org/lindt/custom_datatypes#> .
@prefix geow3c: <http://www.w3.org/2003/01/geo/wgs84_pos#> .
@prefix sosa: <http://www.w3.org/ns/sosa/> .
@prefix ssn: <http://www.w3.org/ns/ssn/> .
@prefix dct: <http://purl.org/dc/terms/> .
@prefix ucum: <http://unitsofmeasure.org/ucum#> .

<observation/235727> a sosa:Observation ;
  sosa:isResultOf <sampling/4578> ;
  sosa:procedure <procedure/134g1> ;
  sosa:hasSimpleResult "22.4 kWh"^^cdt:ucum ;
  sosa:madeBySensor <sensor/927> ;
  sosa:observedProperty <electricConsumption> ;
  sosa:hasFeatureOfInterest <house/134/kitchen> ;
  sosa:isSampleOf <house/134/garden> ;
  sosa:resultTime "2017-12-04T08:14:00:00+10:00"^^xsd:dateTime ;
  sosa:hasFeatureOfInterest <house/134/bedroom> ;
  sosa:isSampleOf <house/134/bedroom> ;
  sosa:madeBySensor <trowel> ;
  sosa:Sampler <trowel> ;
  sosa:Sampling <sampling/4578> ;
  sosa:Sample <sample/134g1> ;
  sosa:hasSample <sample/134g1> .
```

### Listing 3.2: Sampling Modelling

```
@prefix geow3c: <http://www.w3.org/2003/01/geo/wgs84_pos#> .
<http://example.org/> BASE <http://example.org/> .
@prefix cdt: <http://purl.org/dc/terms/> .
@prefix sosa: <http://www.w3.org/ns/sosa/> .
@prefix ssn: <http://www.w3.org/ns/ssn/> .
@prefix ucum: <http://unitsofmeasure.org/ucum#> .

<observation/235727> a sosa:Observation ;
  sosa:isResultOf <sampling/4578> ;
  sosa:madeBySensor <sensor/927> ;
  sosa:observedProperty <electricConsumption> ;
  sosa:isSampleOf <house/134/garden> ;
  sosa:hasFeatureOfInterest <house/134/bedroom> ;
  sosa:resultTime "2017-12-04T08:14:00:00+10:00"^^xsd:dateTime ;
  sosa:hasFeatureOfInterest <house/134/bedroom> ;
  sosa:hasFeatureOfInterest <house/134/garden> ;
  sosa:madeBySampler <trowel> ;
  sosa:Sampler <trowel> ;
  sosa:Sampling <sampling/4578> ;
  sosa:Sample <sample/134g1> .
```

### Listing 3.3: Result Sample Modelling

```
<sample/134g1> a sosa:Sample ;
  sosa:iid "house/100" ;
  sosa:isSampleOf <house/134/bedroom> ;
  sosa:isResultOf <sampling/4578> .
```

3 The complete ontology file for the example is available at: https://github.com/w3c/ssn/blob/master/examples/house134.ttl.
4 The following prefixes are used in the text - `sosa:` for SOSA; `ssn:` for SSN; and `oldssn:` for the old SSNX ontology which defined resources in the namespace http://purl.oclc.org/NET/ssnx/ssn#.
3.3. Actuators and Actuations

As outlined above, there is an increasing importance of the Web of Things and smart instrumentation and environments more generally. Requirements for SSN included one to model actuations (“It should be possible to model actuation functions of sensing devices” [32, §5.27]). An actuation denotes a device’s ability to change something in its environment upon receiving a signal. The SOSA ontology pattern is therefore extended to model `sosa:Actuators`, making some `sosa:Actuations` on some `sosa:ActuableProperty` of a `sosa:FeatureOfInterest`.

For example, Listing 3.5 describes how actuation #188 has been made by actuator windowCloser #987 to act on the state of window #104.

```
Listing 3.5: Actuation Modelling

<actuation/188> a sosa:Actuation ;
  sosa:actsOnProperty <window/104#state> ;
  sosa:madeActuationBy <windowCloser/987> ;
  sosa:resultTime "2017-04-18T17:24:00+02:00"^^xsd:dateTimeStamp.
```

3.4. Spatial aspects

Spatial aspects of an observation, act of sampling or actuation can be associated with the `sosa:FeatureOfInterest`, the `ssn:System` (i.e. the `sosa:Sensor`, `sosa:Sampler` or `sosa:Actuator`) and/or the `sosa:Platform` on which they are mounted. Location may also constitute a `sosa:ObservableProperty` of an observation, for example using a GPS sensor.
Location and other spatial properties of these entities may also be defined according to application- or community-specific models, which in turn make use of appropriate geospatial ontologies such as GeoSPARQL [47].

4. Modularization

Practitioners using the SSNX ontology have identified the complexity of both the ontology and its documentation as a significant usability issue. For example, SSNX imported the Dolce-Ultralite upper ontology (DUL) [45] and many SSNX terms inherited semantics from their parent DUL terms. The DUL alignment contributed a level of complexity and abstraction which affected adoption in some communities. More generally, the monolithic ontology design of SSNX that introduces all terms within one ontology while also relying on the import of the Dolce-Ultralite ontology, makes it difficult for users to focus on just those entities needed for a particular implementation. In response to this, the new SSN ontology is factored into several ontology subsets or modules that are similar in their domain of discourse and directly import one another as needed but differ in their ontological commitments and/or scope of coverage in order to suit different use cases and target audiences. The core module in particular, SOSA, is intended for data repositories and websites managed by web or data managers who need neither extensive axiomatization nor more specialized entities.

Modularization has been accomplished by way of two types or directions of ontology segmentation as shown in Fig. 3.

4.1. Vertical Segmentation

Vertically segmented SSN modules add higher levels of ontological commitment by directly importing lower modules and defining new axioms. The lower level modules are independent of the higher level modules, and logically consistent by themselves. The core SOSA module defines the key classes and properties but axiomatization is deliberately limited. In particular, SOSA uses no object property *rdfs:domain* or *rdfs:range* axioms. In place of these, schema.org *schema:domainIncludes* and *schema:rangelIncludes* annotations provide informal semantics to SOSA properties.\(^5\) SOSA also avoids any subclass and subproperty axioms. The motivation for these choices is to make SOSA simple enough to be incorporated as is in the schema.org extension for IoT.\(^6\) SOSA as the core, does not import any other ontologies, so is truly independent of vertical modules that add more terms, expressivity, and further ontological commitments to the lightweight semantics of SOSA.

The SSN module imports SOSA, and adds full axiomatization to the SOSA classes and properties, including *rdfs:subClassOf*, *rdfs:subPropertyOf*, *owl:disjointWith* and OWL cardinality and guarded existential restrictions on a class-level, along with some further classes and properties to complete the basic conceptual model corresponding approximately to the scope offered by SSNX.

In line with the changes implemented for the new SSN, alignment between SSN terms and DUL terms was reconsidered and externalized to an optional vertical module called the Dolce-Ultralite Alignment module (SSN-DUL) that is available at [http://www.w3.org/ns/ssn/dul](http://www.w3.org/ns/ssn/dul). Those axioms in SSN that include DUL terms have been separated into the SSN-DUL module.

---

5See [https://www.w3.org/2015/spatial/wiki/Domain,_Range, _InverseOf](https://www.w3.org/2015/spatial/wiki/Domain,_Range, _InverseOf) for pointers to the group discussions on this topic.

6The Spatial Data on the Web group endorses SOSA being taken up by schema.org, see thread [https://lists.w3.org/Archives/Public/public-sdw-wg/2017Jun/0067.html](https://lists.w3.org/Archives/Public/public-sdw-wg/2017Jun/0067.html)
ule. The Dolce-UltraLite Alignment Module imports the Dolce-UltraLite Ontology and the Sample relations module, which in turn imports the SSN Ontology, and thereby transitively the SOSA Ontology. However, neither SOSA nor SSN import the Dolce-UltraLite Alignment Module in reverse. The SSN-DUL alignment module has been defined for backwards-compatibility reasons only and SSN (and SOSA) can now be used entirely independently of DUL. In fact, we are not anticipating the use of this alignment in applications other than the one’s that are migrated from the old ontology.

Additional vertical modules (see Sec. 7 axiomatize alignments with other related ontologies, including O&M [13, 26], OBOE [43] and PROV-O [34].

4.2. Horizontal Segmentation

Modules that are horizontally layered may depend on each other, i.e., they may rely on the import of another horizontal module, but only extend scope by adding classes and properties, and do not otherwise enrich the semantics of existing terms. Two horizontal modules are defined in SSN (see Sec. 6.1): the Sample Relations Module and the System Capabilities Module.

5. The New SSN Ontology and its Core

SSNX was developed with ontology engineers as the primary audience in mind. Due to the widespread adoption of SSNX, the increasing role of citizen science, the strong focus on lightweight vocabularies by the Linked Data community, and the rising importance of simple vocabularies such as Schema.org, the SSNX ontology needed streamlining. The new SSN has been modularized so that the core ontology module, SOSA, can be used as a standalone ontology or even a simple vocabulary targeting web developers, citizen science, lightweight Linked Data publishing, resource-constrained IoT devices, and data intensive applications.

SOSA consists of 13 classes, 21 object properties and 2 datatype properties, and includes very little axiomatization. The SSN ontology is available at http://www.w3.org/ns/ssn/. It imports the SOSA ontology, adds additional axioms and 6 classes and 15 object properties. The DL expressivity of the new lightweight SOSA core module is $\mathcal{ALC}^i(D)$ which is efficiently supported by modern DL or RDFS+ reasoners [33, 46, 51], while that of the new SSN is $\mathcal{ACRIN}(D)$. In contrast, the expressivity of SSNX is $\mathcal{SRIQ}$. Table 1 provides some key facts to compare SOSA and SSN to SSNX.

<table>
<thead>
<tr>
<th>Ontology</th>
<th>SOSA</th>
<th>SSN</th>
<th>SSNX</th>
</tr>
</thead>
<tbody>
<tr>
<td>Classes</td>
<td>13</td>
<td>6</td>
<td>41</td>
</tr>
<tr>
<td>Object props</td>
<td>21</td>
<td>15</td>
<td>39</td>
</tr>
<tr>
<td>Datatype props</td>
<td>2</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Tot. logical axioms</td>
<td>13</td>
<td>127</td>
<td>1011</td>
</tr>
<tr>
<td>DL expressivity</td>
<td>$\mathcal{ALI}(D)$</td>
<td>$\mathcal{ACRIN}(D)$</td>
<td>$\mathcal{SRIQ}$</td>
</tr>
</tbody>
</table>

Table 1

Number of classes, object properties, and datatype properties, defined in SOSA, SSN, and SSNX (for comparison); DL expressivity of these ontologies.

Terms defined by the core SOSA ontology are identified by URIs under the namespace http://www.w3.org/ns/sosa/, while those defined by the SSN ontology are identified by URIs under the namespace http://www.w3.org/ns/ssn/. In the rest of this article, we shorten these namespaces with the following prefixes (that are registered at http://prefix.cc).

@prefix sosa: <http://www.w3.org/ns/sosa/> .
@prefix ssn: <http://www.w3.org/ns/ssn/> .

SSN (with its imported SOSA core) is organized, conceptually, but not physically, into eight perspectives. The core component with its three conceptual perspectives (Observations, Sampling, Actuation) has already been described in Sec. 3. In the rest of this section we describe the other conceptual components, how they evolved from SSNX and the main rationale for this evolution. Table 2 in Appendix A lists all of the terms defined in Sosa and SSN, and the terms in the old SSN that they supersede, if they exist.

5.1. Feature of Interest and Sample

In an ontology that aims to describe interactions between the physical and digital world, the target object in the physical world is of primary concern. Even though it is quite common in practice to report measurements of, and actions on, the physical world without explicit reference to a specific target object, the object is still necessarily there, cognitively completing the act of observation. In SSN the target object is called
a sosa:FeatureOfInterest, which reflects a terminology heritage from OGC\(^8\).

The property sosa:hasFeatureOfInterest is used to link between a sosa:Observation and its associated sosa:FeatureOfInterest. This replaces oldssn:featureOfInterest, since the concept oldssn:FeatureOfInterest and the property oldssn:featureOfInterest were distinguished only by case. The new SSN avoids this to make it easier to be used in languages without case distinction.

In the broader context of Spatial Data on the Web, any Spatial Thing could also be a sosa:FeatureOfInterest \([52]\) of an observation, sampling or actuation, as Listing 5.1 exemplifies.

Listing 5.1: Spatial Features of Interest Modelling

```cypher
<house/134 > a sosa:FeatureOfInterest ;
  rdfs:comment "House #134."@en .
<floralCouch> a sosa:FeatureOfInterest ;
  rdfs:comment "The lumpy 2-seater couch with the floral pattern."@en .
<roofInsulation> a sosa:FeatureOfInterest ;
  rdfs:comment "The insulation material in the roof of house #134."@en .
```

SSN defines sosa:Sample as a subclass of sosa:FeatureOfInterest, because, when used for an observation, the sample is the (proximate) feature of interest, as Listing 5.2 shows.

Listing 5.2: Proximate Feature of Interest Modelling

```cypher
<observation/s1/5> a sosa:Observation ;
  sosa:hasFeatureOfInterest <sample/134g1> ;
  sosa:observedProperty <soil-pH> ;
  sosa:phenomenonTime [ a time:Instant ;
    time:inXSDDateTimeStamp "2017-12-04T08:14:00+10:00"^^xsd:dateTimeStamp ] ;
  sosa:resultTime "2017-12-04T10:24:00+10:00"^^xsd:dateTime ;
  sosa:hasSimpleResult "6.1"^^cdt:pH .
```

In most circumstances the sample is only of interest in the context of observations, because it represents the (ultimate) feature of interest that is the object of real world or scientific interest within the investigation. In any observation we can query the property path\(^9\) sosa:hasFeatureOfInterest/sosa:isSampleOf to find the ultimate feature of interest for the observation. In Listing 5.2, which continues the example introduced in Sec. 3, we can infer that the ultimate feature of interest is in fact the garden denoted `<house/134/garden>`.

A sosa:FeatureOfInterest and sosa:Sample will often have a specified location or other spatial properties, but this is not required. Observations can be made and samples taken from features for which the location is of no direct interest. For example, a material sample may represent a substance; an individual organism may be representative of a taxon; a statistical sample may represent a particular community that is not characterized by location, etc.

5.2. Properties

An observation targets a feature of interest but interacts with it only through estimating the value of a characteristic or property of that feature. The general class of feature properties, ssn:Property, is defined in the SSN module, complementing its subclasses in SOSA, sosa:ObservableProperty and sosa:ActuableProperty. The ssn:hasProperty, and its inverse ssn:isPropertyOf, relate a feature of interest to ssn:Property. We can state, as shown in Listing 5.3, that air temperature is a property of each room in the house.

Listing 5.3: Property Modelling

```cypher
<airTemperature> a ssn:Property ;
  ssn:isPropertyOf <house/134/bedroom>,
  <house/134/kitchen> .
```

Some properties of a feature may not be observable or actuable characteristics, such as ‘name’. SOSA/SSN focus on those feature properties that have a triple connection to observation or actuation events: as an ssn:Property of the sosa:FeatureOfInterest of a sosa:Observation or sosa:Actuation, as the property being observed or actuated, and as the property usually observed or actuated by the sensor or actuator used in the observation or actuation. Any of these property paths should be able to be inferred from the others, whether or not all are explicitly stated.

5.2.1. Observable Properties

The multiplicity of property paths in which sosa:observedProperty is involved can lead to significant modeling and representation choices. On the one hand, as an inherent characteristic, a sosa:observedProperty is specific to a feature. On the

\(^8\)The geospatial community uses the term ‘feature’ primarily to refer to discernable, identifiable objects in the landscape and their digital representations.

\(^9\)See https://www.w3.org/TR/sparql11-query/#propertypaths for the definition of SPARQL property paths.
other hand, we consider that multiple observations across different features of interest or by different sensors or both can measure the same generic property, e.g., air temperature. Thus, the air temperature in different rooms of a smart home might be observed as shown in Listing 5.4.

Listing 5.4: Generic Observable Property Modelling

```xml
<observation/235714> a sosa:Observation ;
  sosa:hasFeatureOfInterest <house/134/bedroom> ;
  sosa:observedProperty <airTemperature> ;
  sosa:madeBySensor <sensor/926> .
<observation/235728> a sosa:Observation ;
  sosa:hasFeatureOfInterest <house/134/kitchen> ;
  sosa:observedProperty <airTemperature> ;
  sosa:madeBySensor <DHT22/2> .
```

It might even be that the same digital portable thermometer is being used to measure both air and water temperature (considered as an even more generic Ambient Temperature) alternately in both rooms.

The link from a sensor to the property that it observes, is made using the sosa:observes property, implying that every observation involving this sensor is about the same property. In Listing 5.5, sensor #926 is deployed only to observe the air temperature in the bedroom of our home, and we know that it made some observations.

Listing 5.5: Observable Property Modelling

```xml
<sensor/926> a sosa:Sensor ;
  sosa:observes <airTemperature> ;
  sosa:madeObservation <observation/235714>,
    <observation/235715>, <observation/235716> .
```

Since <airTemperature> is a generic Property and not specific to the bedroom, we cannot just follow the sensor property path and infer which room’s air temperature was being observed: we need to know the sosa:hasFeatureOfInterest as well. A specific Property could instead be used, such as <airTemperature>House/134/bedroom>, but it would then be more difficult either to express the capabilities of a portable sensor or to represent the generalization <airTemperature>.

One approach to modeling generalization relationships between individual properties might be to use the qudt:specialization and qudt:generalization properties from the Quantities, Units, Dimensions and Data Types vocabulary [25]. For example, various temperatures around the home may be modeled as in Listing 5.6.

Listing 5.6: Generalised Property Modelling

```xml
@prefix qudt: <http://qudt.org/schema/qudt#> .
<temperature> a sosa:ObservableProperty,
  qudt:QuantityKind ;
  rdfs:label "Thermodynamic Temperature"@en .
<airTemperature> a sosa:ObservableProperty,
  qudt:QuantityKind ;
  qudt:generalization <temperature> .
<soilTemperature> a sosa:ObservableProperty,
  qudt:QuantityKind ;
  qudt:generalization <temperature> .
<waterTemperature> a sosa:ObservableProperty,
  qudt:QuantityKind ;
  qudt:generalization <temperature> .
<tapWaterTemperature> a sosa:ObservableProperty,
  qudt:QuantityKind ;
  qudt:generalization <waterTemperature> .
<poolWaterTemperature> a sosa:ObservableProperty,
  qudt:QuantityKind ;
  qudt:generalization <waterTemperature> .
```

Some ontology engineers (and some controlled vocabularies for quantity kinds) prefer to model such generalization-specialization relations using sub-class relations between observable property classes. In SSNX, for example, the class of observable properties was a subclass of dul:Quality and types of properties were usually modeled as subclasses of oldssn:ObservedProperty rather than as individuals.10 Using either approach in SSN, it is still necessary to create an individual of whatever observed property class(es) is appropriate, whether that individual is still a generic property or is specific to an individual feature of interest as shown in Listing 5.7 or Listing 5.8.

Listing 5.7: Specific Property Modelling

```xml
<AmbientTemperature> rdfs:subClassOf sosa:ObservableProperty .
<AtmosphericTemperature> rdfs:subClassOf AmbientTemperature> .
```

10In SSNX, the class of observable properties was defined as a subclass of dul:Quality and as the class of “observable 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.” Section 5.3.1.3.4 in [36] further adds that types of properties, such as temperature or pressure should be added as subclasses of oldssn:ObservedProperty instead of individuals. Yet, usage reports [19] of SSNX revealed that most datasets were using observable properties as applicable to many features of interest. See https://www.w3.org/2015/spatial/wiki/What_is_an_instance_of_ssn:Property for more details on this point.
5.2.3. Actuatable Properties

Sensors respond to stimuli, e.g., changes in the environment, or input data composed from the results of prior observations, to generate the result. The class ssn:Stimulus is unchanged from SSNX and is a proxy (i.e., ssn:isProxyFor) for an observable property, or a number of observable properties. For example, the amount of dynamic acceleration in an accelerometer as a proxy for the tilt angle of a smartphone. Properties themselves are observable characteristics of (i.e., ssn:isPropertyOf) real-world entities (i.e., ssn:FeatureOfInterests).

5.2.3. Actuatable Properties

The class of properties that can be acted on by actuators is sosa:ActuatableProperty. Instances of sosa:ActuatableProperty are usually generic to many features of interest (e.g., <openCloseStatus>), but they may be defined as specific to a single feature of interest (e.g., <window/104#state>).

For example, using a feature of interest - generic modeling choice for the previous example introduced in Listing 3.5, one may model the open or close status of the window as a first-class citizen as described in Listing 5.9.

Listing 5.9: Generic Feature of Interest Modelling

```owl
<openCloseStatus> a <ssn:FeatureOfInterests> ;
ssn:forProperty <window> ;
ssn:hasInput <electricityConsumption> ;
ssn:hasOutput <windowOpenCloseStatus> .
```

This change addressed several requirements (e.g., [32, §5.25] and [32, §5.40]) and is the result of extensive discussions within the group. The sosa:Procedure class now allows users to describe the protocol of Web services which themselves may represent ways on how to interact with, for example, sosa:Actuators. In particular, a sosa:Procedure can be linked to some description of the ssn:Input and ssn:Output of these procedures. For example, Listing 5.10 states that the output of the procedure is the electricity consumption.

Listing 5.10: Output Modelling

```owl
<electricityConsumption> a <ssn:ObservableProperty> ;
sosa:hasOutput <electricityConsumption> .
```

5.3. Procedures and their Specification

Conceptually, an observation, sampling, or actuation, can be thought of acts that use(der) (parts) of a procedure. A sosa:Procedure is a reusable workflow, protocol, plan, algorithm, or computational method that can be used to specify how to make an observation, create a sample, or make a change to the state of the world (i.e., perform an actuation). This represents a significant departure from the semantics of the oldssn:Process class, defined as a subclass of dul:Process, and which described the real-world process, rather than a description of it. In the optional DUL alignment, a sosa:Procedure is now defined as subclass of dul:Method to describe a workflow that specifies how to make an observation, create a sample, or make a change to the state of the world via an actuator (see Sec. 5.3). The oldssn:Sensing class, which was defined as a subclass of a oldssn:Process has seen very little implementation evidence and has been deprecated in SSN.

This change addressed several requirements (e.g., [32, §5.25] and [32, §5.40]) and is the result of extensive discussions within the group. The sosa:Procedure class now allows users to describe the protocol of Web services which themselves may represent ways on how to interact with, for example, sosa:Actuators. In particular, a sosa:Procedure can be linked to some description of the ssn:Input and ssn:Output of these procedures. For example, Listing 5.10 states that the output of the procedure is the electricity consumption.
If the result of an observation is a web document having some representation (e.g., in JSON or XML) other ontologies such as the RDF Presentation ontology\textsuperscript{12} [37] or RaUL [23] can be used to model the result using the `ssn:hasOutput` relation of SSN. It links the output of an observation to a description of how a result document can be validated (e.g., using some JSON Schema), or how an RDF description can be obtained from it (using some RDF lifting rule such as [39, 48] as shown in Listing 5.11).

Listing 5.11: Validation Rule Modelling

```rdf
@prefix rdfp: <https://w3id.org/rdfp/> .

<summingHourlyConsumptionProcedure> a sosa:Procedure ;
<observation/235727> a sosa:Observation ;
 sosa:hasResult <observation/235727/result> .
```

Procedures linked to observation and sampling activities, beyond the description of the presentation of the output, are typically a record of how these activities are performed (a log), and are linked to a `sosa:Sensor` or `sosa:Actuator` with the `ssn:implementedBy` relation as described in Listing 5.12.

Listing 5.12: Procedure Modelling

```rdf
<sensor/927> a sosa:Sensor ;
 ssn:implement <summingHourlyConsumptionProcedure> .
```

Procedures linked to actuation activities, however, can be either 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 `ssn:System` concept and its relations to activities through `ssn:implementedBy/implements` in combination with the inputs and outputs of a procedure can define the interface of how to interact with a `sosa:Actuator`. 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 SSN. Existing ontologies such as OWL-S [44] and execution protocols such as WSMX [21] can be used together with lower-level specifications such as the W3C Thing Description\textsuperscript{13} to model these details.

5.4. Results

SSN offers two ways of attaching properties to activities that observe, actuate or sample them. For simple (though frequent) cases that merely require a literal typed with an appropriate datatype, one may use the `sosa:hasSimpleResult` datatype property. Alternatively, observation results can be modeled as individuals and linked to the observation using the object property `sosa:hasResult`. In cases where the observed property is a physical quantity, one may then use one of the several existing ontologies to model the result.

5.4.1. hasSimpleResult and hasResult

Listing 5.13 shows how to attach a literal value to our previous `observation/235715` object. It also uses a custom datatype (i.e., `cdt:temperature`) whose value space is some set of quantity values. However, such custom datatypes are not compatible with the OWL specifications, which restricts the set of datatypes that can be used and may consequently lead to an error in the OWL reasoner.

Listing 5.13: Simple Result Modelling

```rdf
<observation/235715> a sosa:Observation ;
 sosa:resultTime "2017-11-15T14:35:13Z"^^xsd:dateTime ;
 sosa:hasSimpleResult "23.9 DEG"^^cdt:temperature .
```

The `ssn:hasResult` object property allows us to make statements about the `ssn:Result` object (as shown in Listing 5.14) by explicitly stating the unit of measurement for the value of the observation. Although it was not in the scope of the SSN specification to recommend any particular way of modeling results as quantity values, there exist several external vocabularies that are specifically designed for modeling quantity values as OWL individuals. Examples include the QUDT ontology ([25]) used in this listing, or the Ontology of Units of Measure (OM [49]). With QUDT, a `sosa:Result` would be a `qudt:QuantityValue`. With OM, a `sosa:Result` would be an `om:Measure`.

Listing 5.14: Qualified Result Modelling

```rdf
<observation/235716> a sosa:Observation ;
 sosa:resultTime "2017-11-15T14:35:13Z"^^xsd:dateTime ;
 sosa:hasResult [ a qudt:1-1:QuantityValue ;
 qudt-1-1:unit qudt-unit-1-1:DegreeCelsius ;
 qudt-1-1:numericValue "23.9"^^xsd:double ] .
```

The two alternative patterns reduce the complexity of the path to link an observation to the actual literal that encodes some value for the

\textsuperscript{12}https://w3id.org/rdfp/
\textsuperscript{13}https://w3c.github.io/wot-thing-description/
result of this observation that was required in SSNX\textsuperscript{14}. Compatibility with SSNX (see Sec. 7.4) is obtained through both oldssn:SensorOutput and oldssn:ObservationValue being subclasses of the new sosa:Result, and oldssn:observationResult being a subproperty of ssn:hasResult.

There have been discussions in the working group on potential actuation commands\textsuperscript{15}, but their inclusion has been deemed out of scope for the current version of SSN.

5.4.2. Result of a Sampling activity

In SSNA, a subclass of sosa:Result because a sample is the result of a sosa:Sampling activity. This is in addition to being a subclass of sosa:FeatureOfInterest so that it can serve as the target of a sosa:Observation.\textsuperscript{16}

5.4.3. Result Time and Phenomenon Time

SOSA distinguishes between the sosa:phenomenonTime and sosa:resultTime, the former being the time that the result of an observation, actuation, or sampling applies to the feature of interest, while the latter specifies the instant of time when an act, such as an observation, was completed.

The sosa:resultTime is a datatype property with rdfs:range xsd:dateTime to capture the time instant that the activity completed and the result obtained. The sosa:phenomenonTime is an object property with range time:TemporalEntity \textsuperscript{15}, since it may be either an instant or interval, or even a temporal complex.

For example, Listing 5.15 describes that the temperature was observed through April 15th 2017, and that the result was available 12 seconds after the end of this period.

Listing 5.15: Phenomenon Time Modelling

```xml
<observation/235714>
  sosa:resultTime "2017-04-16T00:00:12+00:00"^^xsd:dateTimeStamp ;
  sosa:phenomenonTime [ a time:Interval ;
    time:hasBeginning [ a time:Instant ;
      time:inXSDDateTimeStamp "2017-04-16T00:00:00+00:00"^^xsd:dateTimeStamp
    ] .
  ] .
</observation/235714>
```

The distinction between sosa:resultTime and sosa:phenomenonTime is important where the end of the observation, actuation, or sampling, activity is significantly different from that of the phenomenon that is observed or acted on. The separation covers cases involving observations over a long period of time, late results due to long-lasting procedures, delays due to lengthy information travel (e.g., information traveling long distances), cases where the result describes the state in the distant past (e.g., observations that described the pressure and temperature of a mineral at its time of formation), and predictions and forecasts - which may be defined as observations with a sosa:resultTime before the sosa:phenomenonTime [13, 22, 26, §7.2]. In both of the last two cases, it is common to have multiple observations relating to the same feature of interest, observed property and phenomenon time, but with different result times. For example, different procedures might be used in different geology labs. And the results of forecasts obtained using computational procedures might later be subject to validation by instrumental observations. The outcome of the latter would be encoded as sosa:Observations with the same sosa:hasFeatureOfInterest, sosa:observedProperty, and sosa:phenomenonTime, as the forecast, but with a different sosa:usedProcedure, sosa:madeBySensor and sosa:resultTime.

5.5. Systems and their deployments

The modelling of a sensor as a physical piece of technology and the way multiple sensors are attached to other such entities has been greatly simplified in SSN. Whereas SSNX distinguished between an oldssn:Sensor that could be any entity that performed oldssn:Sensing and an oldssn:SensingDevice that was a subclass of an oldssn:Device, i.e. a physical piece of technology, the new SSN drops the notion of an oldssn:Device as well as the notion of an oldssn:SensingDevice. The ssn:System class that already existed in SSNX takes the place of those two classes, i.e. it can be used if someone wants to explicitly model that a sosa:Sensor, sosa:Actuator or sosa:Sampler is a physical piece of technology. For

\textsuperscript{14} Wiki page https://www.w3.org/2015/spatial/wiki/Storing_Observation_Value lists pros and cons on the options that were considered, and excerpts of the discussion within the group.

\textsuperscript{15} See https://www.w3.org/2015/spatial/wiki/Result_and_Command for a summary of the discussion on this topic.
similar reasons, the notion of a sampling device has been dropped\(^\text{16}\).

5.5.1. Platforms and hosts

One or more sensors (as well as actuators and samplers) can be hosted or mounted on a \texttt{sosa:Platform}. Such platforms can also define the geometric properties, i.e., placement, of sensors in relation to one another. \texttt{sosa:Platforms} can also host other \texttt{sosa:Platforms}.

The properties \texttt{oldssn:attachedSystem} and \texttt{oldssn:onPlatform} have been renamed to \texttt{sosa:hosts} and \texttt{sosa:isHostedBy}, respectively, and they can now be used on both, the \texttt{sosa:Platform} and the \texttt{sosa:System} class to define that they host sensors, actuators or samplers.\(^\text{17}\)

Temperature observations can be defined to be made, for example, by a temperature sensor that is built-in/hosted by a Nest thermostat. Listing 5.16 shows how to model this relation in our smart home use case. In this example, \texttt{<sensor/D1AA22A82>} is inferred to be an \texttt{ssn:System}, as only \texttt{ssn:Systems} can be hosted by \texttt{sosa:Platforms}.

Listing 5.16: Platform Modelling

\begin{verbatim}
<nest/D1AA22A8211> a sosa:Platform ;
  rdfs:label "Nest Learning Thermostat D1AA22A8211"@en ;
  rdfs:comment "3rd gen Nest Learning Thermostat D1AA22A8211"@en ;
<sensor/926> a sosa:Sensor ;
  rdfs:label "Nest temperature sensor #1"@en ;
  rdfs:comment "Nest Thermostat in bedroom of house #134"@en ;
  sosa:hosts <sensor/926> .

<house/134/boiler> a sosa:Platform ;
  rdfs:label " Boiler"@en ;
  rdfs:comment "Boiler in boiler room of house #134"@en ;
  sosa:hosts <PCBBoard1> .

<PCBBoard1> a ssn:System ;
  rdfs:label "PCB Board 1"@en ;
  rdfs:comment "PCB Board 1 hosts DHT22 temperature sensor #1"@en ;
  sosa:hosts <PCBBoard1> .

<PCBBoard2> a ssn:System ;
  rdfs:label "PCB Board 2"@en ;
  rdfs:comment "PCB Board 2 hosts DHT22 temperature sensor #2"@en ;
  sosa:hosts <PCBBoard2> .

<house/134/deployment> a ssn:Deployment ;
  rdfs:label "Deployment of PCB Board 1 and 2 in the kitchen for the purpose of observing the temperature"@en ;
  sosa:deployedSystem <PCBBoard1>, <PCBBoard2> ;
  sosa:hosts <house/134/kitchen> ;

<house/134/deployment> a sosa:Platform ;
  rdfs:label "Deployment of PCB Board 1 and 2 in the kitchen for the purpose of observing the temperature"@en ;
  sosa:hosts <PCBBoard1>, <PCBBoard2> .

Listing 5.17: Deployment Modelling

\end{verbatim}

5.5.2. Systems and Sub-systems

The system class has remained unchanged from SSNX apart from its relation to DUL (see Sec. 3.1) as a unit of abstraction for pieces of infrastructure that implement procedures and that are hosted by platforms. A system can have components \texttt{(ssn:hasSubSystem)} which are also systems. Classes and relationships related to system capabilities, operating ranges, and survival ranges, under given conditions have been relegated to a separate horizontal module (see Sec. 6.1).

5.5.3. Deployment

An \texttt{ssn:Deployment} is an activity or a set of activities that encompass all phases in the lifecycle of a deployed system, such as, the installation, maintenance and decommissioning of the platform, sensors, actuators or samplers attached to that system. The class \texttt{ssn:Deployment} describes the deployment of one or more systems \texttt{(ssn:deployedSystem)} or platforms \texttt{(ssn:deployedOnPlatform)} for a particular purpose. For example, a temperature sensor deployed on a wall, or a whole network of sensors deployed for an observation campaign.

Listing 5.17 shows an example of how to model a deployment (i.e. \texttt{<house/134/deployment>}) of two systems \texttt{<PCBBoard1>}, \texttt{<PCBBoard2>} that are deployed in the kitchen of our example house \texttt{#134}.

6. Horizontal Extension Modules

Horizontal extension modules are ontologies that introduce new classes and relations on top of SSN.

6.1. The System Capabilities Module

The System Capabilities module, a.k.a. SSN-System, gathers classes and properties used to model
system capabilities, operating range, and survival range. This part of SSNX was rarely used in implementations, hence specific effort has been made to clarify and simplify its documentation, along with providing illustrative examples. Terms defined by the SSN-System ontology are identified by URIs under the namespace http://www.w3.org/ns/ssn/systems/. We shorten this namespace with prefix ssn-system

@prefix ssn-system: <http://www.w3.org/ns/ssn/systems/> .

Table 3 in Appendix B lists all of the terms defined in SSN-System, and the terms in the old SSN that they supersede, if they exist.

6.1.1. System Capabilities

An ssn-system:System may be linked to some ssn-system:SystemCapability, which in turn is linked to some ssn-system:SystemProperty: measurement, actuation, or sampling properties that describe the capabilities of the ssn-system:Such as its accuracy, latency, precision, etc. An ssn-system:SystemCapability can furthermore be linked to some ssn-system:Condition that define its validity context such as a temperature range. For example, Listing 6.1 specifies that the DHT22 temperature sensor sensitivity is 0.1°C in normal temperature conditions.

Listing 6.1: System Capability Modelling

```xml
@prefix schema: <http://schema.org/> .
@prefix qudt-unit-1-1: <http://qudt.org/1.1/vocab/unit#> .

<DHT22TempSensitivity> a ssn:Property ;
    rdfs:comment "The sensitivity and resolution of the temperature sensor is 0.1 °C in normal temperature and humidity conditions."@en ;
    schema:PropertyValue ;
    schema:unitCode qudt-unit-1-1:DegreeCelsius .

The Terms ssn-system:SystemCapability and ssn-system:SystemProperty are named after the SSNX oldssn:SensorCapability and oldssn:SensorProperty terms, that were generalized so that the new ones apply to the more general concept of ssn:System instead of just sosa:Sensor.\(^\text{18}\) Several sub-classes of ssn-system:SystemProperty are pre-defined as listed in Table 3. Most of these were already defined in SSNX but their definition was harmonized, simplified, and sometimes generalized when the term can be applied equally to a sosa:Sensor, sosa:Actuator or sosa:Sampler. A system property ssn-system:ActuationRange has been introduced as it was considered of high interest for actuators.

6.1.2. Operating Range and Survival Range

The pattern used to describe a system capability in terms of some of its property values under certain conditions is replicated to describe its operating range and survival range. An ssn-system:OperatingRange (resp. ssn-system:SurvivalRange) describes some normal ssn-system:OperatingProperty (resp. ssn-system:SurvivalProperty) of an ssn-system:Under some specified ssn-system:Conditions. For example, the power requirement or maintenance schedule of an ssn-system:Under the system or its battery lifetime) under a specified temperature range. In the absence of an ssn-system:OperatingProperty (an ssn-system:SurvivalProperty, it simply describes the ssn-system:Conditions in which a System is expected to operate (under which the ssn-system:Can be exposed to without damage). The ssn-system:Continues to operate as defined by its ssn-system:SystemCapability. If, however, the ssn-system:OperatingProperty (resp. ssn-system:SurvivalProperty) is violated, the ssn-system:Will be in the operating range’ (resp. is ‘damaged’) and its ssn-system:SystemCapability specification may no longer hold. Some sub-classes of ssn-system:OperatingProperty and ssn-system:SurvivalProperty are also pre-defined, and listed in Table 3.

\(^{18}\)See https://www.w3.org/2015/spatial/wiki/Measurement_and_Operating_properties_for_actuators and https://lists.w3.org/Archives/Public/public-sdw-wg/2017Mar/0233.html for more details on the discussions.
6.1.3. Extensibility of the System Capabilities Module

As a matter of fact, the SSN System Capabilities module contains a predefined list of capabilities, operating ranges, and survival ranges, that were considered of high relevance for SOSA/SSN users. External ontologies may propose new such terms in their own namespace, or even reuse the design pattern to describe other characteristics of other kinds of systems (for example, the maximal bandwidth or payload of a communicating device in some conditions).

6.2. The Sample Relations Module

Support for samples and sampling is one of the major enhancements in SOSA/SSN. The defining property of a `sosa:Sample` is the `sosa:isSampleOf` relationship with the thing that it is intended to represent.

However, in many cases a sample also has a relationship with another sample or samples as part of a study or deployment [26, 49][32, §5.38]. The nature of the relationship is typically quite specific, for example:

- spatial, such as pixels within a remote-sensed scene, stations along a traverse, or specimens collected along a borehole
- specific fractions of a mixture, such as platelets from a blood sample
- specific ("biased") fractions of an assay sample, such as the fraction of a powder that passes a specific sieve grating, the fraction that is magnetically susceptible, or the fraction whose density is higher than a specified value
- non-specific ("un-biased") fractions of a sample ("splits")
- parts of a specimen, such as the leg of an insect, which in turn represents a taxon

The design of sub-sampling strategies is a key element of scientific investigations, and is a critical source of innovation. Therefore, generic relationships such as ‘parent-child’, or ‘subset’ are insufficient.

The Sample Relations module provides a scalable pattern for linking samples which also allows relationship details to be captured. This is accomplished by introducing an intermediate class in the relationship between samples, so that the nature of the relationship can be recorded as an annotation on the association.

In Listing 6.2, the nature of the relationship between a sub-sample of our soil and the sample within which it is found is described in an `rdfs:comment` within a blank-node.

```rdfs
<sosa:Sample> a sosa:Sample ;
  rdfs:label "Soil sample 134gl organic fraction" ;
  sampling:hasSampleRelationship [a sampling:SampleRelationship ;
    sampling:natureOfRelationship [a sampling:RelationshipNature ;
      rdfs:comment "organic fraction of material" ;
    ] ;
    sampling:relatedSample <sample/134gl> ;
  ] ;
  sosa:isResultOf <sampling/4650> .
</sosa:Sample ;
<sampling/4650> a sosa:Sampling ;
  sosa:featureOfInterest <sample/134gl> ;
  sosa:usedProcedure <procedure/soil-organic-fraction/78> .
</sampling/4650 ;
<procedure/soil-organic-fraction/78> a
  sosa:Procedure ;
  rdfs:comment "... details of procedure to separate the organic fraction of a soil sample ..." .
</procedure/soil-organic-fraction/78>
```

If there is a set of ‘standard’ relationships used within a particular discipline or community, these could be registered and assigned URIs. A reference to the standard relationship can then be used instead of the blank-node, as described in a modified version of the example above in Listing 6.3.

```rdfs
<sosa:Sample> a sosa:Sample ;
  rdfs:label "Soil sample 134gl organic fraction" ;
  sampling:hasSampleRelationship [a sampling:SampleRelationship ;
    sampling:natureOfRelationship <http://soil.example.org/rel/organic-fraction> ;
    sampling:relatedSample <sample/134gl> ;
  ] ;
  sosa:isResultOf <sampling/4650> .
</sosa:Sample ;
<sampling/4650> a sosa:Sampling ;
  sosa:featureOfInterest <sample/134gl> ;
  sosa:usedProcedure <procedure/soil-organic-fraction/78> .
</sampling/4650 ;
<procedure/soil-organic-fraction/78> a
  sosa:Procedure ;
  rdfs:comment "... details of procedure to separate the organic fraction of a soil sample ..." .
</procedure/soil-organic-fraction/78>
```

The Sample Relations module is included in SSN/-SOSA as a non-normative horizontal extension.

7. Vertical Extension Modules

Vertical extension modules are ontologies that introduce additional expressivity or axiomatic constraints on top of SSN.

7.1. Alignment to OGC O&M

The observation and sample aspects of SSN are closely aligned to the OGC O&M model precedent [13, 26]. However, (a) OGC O&M is defined using UML and (b) some of the class and property names have been adjusted. Therefore, a formal (axiomatized)
alignment to OGC O&M is provided as a vertical module which owl:imports SOSA, and uses the URI scheme defined by ISO 19150-2 to denote the O&M classes and properties [27].

The main conceptual difference between SSN and O&M is that the latter conflates both Procedures and Systems into a pair of classes: OM_Process and SF_Process. Alignment is achieved by way of the following axiom: iso19156-om:OM_Process ⊑ sosa:Sensor ∧ sosa-om:ObservationProcedure ⊑ prov:Activity

Some other entities and properties of O&M, such as validTime, are not presently mapped to any terms of SOSA/SSN.

7.2. Alignment to PROV-O

The re-orientation of SSN so that observations (as well as samplings and actuations) are activities or “acts of sensing | sampling | actuation” means that SSN now clearly matches a process-flow model. This makes an alignment with PROV-O [34] quite natural, as foreseen in [10, 14]. A formal (axiomatized) alignment to PROV-O is provided as a vertical module which owl:imports SOSA and PROV-O. The key alignments are:

1. sosa:Observation ⊑ prov:Activity
   sosa:Sampling ⊑ prov:Activity
   sosa:Actuation ⊑ prov:Activity;

2. ssa:System ⊑ prov:Agent ∩ prov:Entity
   sosa:Sensor ⊑ prov:Agent ∩ prov:Entity
   sosa:Sampler ⊑ prov:Agent ∩ prov:Entity

Thus, the alignment expresses this as a property-chain axiom: oboe:measurementFor ⊑ prov:Entity ⊑ sosa:hasFeatureOfInterest.

7.4. Alignment to the Old SSN Ontology

It is strongly recommended that applications that use the SSNX ontology are migrated to the new SSN ontology published at: http://www.w3.org/ns/ssn/. However, for legacy applications that continue to use SSNX, a new version of that ontology has been published at the old namespace: http://purl.oclc.org/NET/ssnx/ssn. This new version has been updated as follows:

- some errors with the previous version, such as the outdated import location for the DUL ontology, have been fixed;
- all axioms that pertain to the alignment of the SSNX to the Dolce-UltraLite ontology have been removed and an import of the DUL alignment module http://www.w3.org/ns/ssn/dul has been added instead;
- and mapping relations to the new SSN through owl:equivalentClass, owl:equivalentProperty and owl:subClassOf axioms have been added, while owl:deprecated annotation properties have been added to terms that were removed in the new SSN.

The June 2011 version of SSNX remains available at https://www.w3.org/2005/Incubator/ssnx/ssnx, where either an HTML (ssn.html) or an RDF/XML (ssn.owl) representation will be served by the W3C server depending on the client preferences.

8. Topics out of Scope for SSN

While the reworked SSN has achieved most of the goals set by the Charter and the Working Group chairs [54], and more, there are some concepts that are relevant to the intended scope of SSN but are not included in any of the present SOSA/SSN modules.

Since the publication of SSNX [36], the authors have been collecting feature requests from the user community. Early in the standardization process the Working Group members were asked to solicit and document use cases from which requirements were derived and published by the W3C and OGC [41]. Of the 28 requirements for SSN noted there, 10 relate to the representation of temporal and spatial aspects of sensors and their observations. Like SSNX before it, a
decision was made to defer to external ontologies for such modelling, and this is consistent with the advice of the Working Group in its Best Practice publication [52], which identifies some suitable ontologies ranging from W3C-BASIC-GEO [7] to GeoSPARQL [47]. Although this omission from SSN will have the effect of making SSN harder to use, and perhaps too lightly passes off demanding requirements for streaming measurements, mobile sensors and compact time series, there is no ideal single answer to this need. These requirements may be well served by followup primer publications currently planned, in preference to ontology extension.

Two requirements are met simply by virtue of the choice of a Web Ontology Model for SSN, and three others have been met by preserving SSNX capabilities. Nine have been met by the new work reported here. Arguably, a requirement for verifiable profiles has been partly met by the modularization presented here, and a requirement for examples in concert with external vocabularies has been met via the ontology mappings and examples presented in the specification and here. Multilingual annotation is being developed at present. The requirement for aggregated observations has not been specifically addressed.

Other informally requested features that have not been implemented include direct support for networks of sensors, such as device communications, network structures, relations between sensors, or specific data discovery and exchange interfaces. Some IoT-specific ontologies such as [3] and [4] (built on SSNX) as well as [17], and APIs such as [40] do cover some of these aspects. The recursive ssn:System class might be a starting point for network nodes, but network relations would still be required. Such relations could perhaps be modelled by extending the concepts InteractionPattern and Link of the “Thing Description” standard being developed by the W3C Web of Things Working Group. The new terms for actuation do not include actuation commands (analogous to sensor capabilities for observations). Other ontologies could add these concepts as a vertical module. For example, the Procedure Execution ontology, one of the core modules of the Smart Energy Aware Systems ontology [38] generalizes the core classes and properties of SSN to describe pep:ProcedureExecutor (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), which may then be linked to some description of the command and/or the result.

9. Evaluation of Implementation Evidence

Since its publication in 2011 the SSNX ontology has been used in many IoT applications, linked datasets and ontologies. Examples of its use in linked datasets include meteorological models from the Spanish Meteorological Office [1], a case study on environmental sensing and livestock monitoring published by CSIRO [53], long-term climate observation data published by the Australian Bureau of Meteorology [35], real-time passenger data in the GetThere smartphone app [11] and fault analysis and worker support for cyber-physical production systems in a case study in the German Industrie 4.0 initiative [56].

The decision to formally standardize the SSN was a consequence of the interest expressed at a public workshop held in London in March 2015 https://www.w3.org/2014/03/lgd/ and was managed through a joint W3C-OGC working group.

The use of SSN for data and applications published openly on the Web has been documented in detail by the Spatial Data on the Web Working Group in a note published at: https://w3c.github.io/sdw/ssn-usage/. The list of usage of SSNX and SSN was obtained by crawling the LOD Laundromat, the LOD Cloud Cache, LODStats and the LOV Ontology repository, as well as through requests for implementation evidence on the Spatial Data on the Web working group mailing list. During the development of the new SSN between March 2016 and October 2017, the working group collected implementation evidence of all old (but equivalent) and new SOSA/SSN terms in the same document [19]. Standardization of the SSN through the W3C required evidence of use of each term in at least two consumer and two provider applications, i.e. applications or datasets that use SSN to describe their data and ontologies that extend SSN, respectively.

At the time of its publication as a W3C candidate recommendation in July 2017, SOSA/SSN terms had been used in 23 open linked datasets, and 23 openly published ontologies were known to use the SSN ontology to either describe their data or extend the SSN ontology, respectively. All terms have been used at least once in a dataset and an ontology, while 87% of
10. Conclusion

The W3C/OGC Spatial Data on the Web Working Group has redesigned the SSNX ontology that was published through a W3C Incubator Group in 2011, incorporating feedback from users of the ontology and extended it with terms to model sampling and actuation activities that have been identified as missing from the original ontology in many use cases. A new version of the ontology has been published as an official W3C Recommendation and an OGC implementation standard at: http://www.w3.org/ns/ssn/.

The initial SSNX was perceived as too heavyweight in its axiomatization, and too dependent on OWL reasoning by some users. To strike a balance, the complexity of the new SSN ontology has been reduced through a modularization that allows different user groups to pick and choose terms of the ontology appropriate for their domain of discourse and also chose between different levels of axiomatization of the ontology.

Another key novelty in the modularization of SSN has been the introduction of the lightweight and self-contained core pattern called SOSA (Sensor, Observation, Sampler, and Actuator) as a replacement of the initial Stimulus Sensor Observation (SSO) pattern which is available at: http://www.w3.org/ns/sosa/. SOSA and SSN together can now be used to describe sensors and their observations, the involved procedures, the studied features of interest, the samples used to do so, the feature’s properties being observed or sampled, as well as actuators and the activities they trigger.

SSNX has already been broadly accepted as a de-facto standard for representing sensor data on the Web and has been the inspiration for multiple ontologies layered on top of SSNX. With the standardization of the SSN and SOSA ontologies through the OGC and the W3C and the introduction of different modules for different audiences, we expect the new ontology to be used in even more varied use cases, especially in the Internet-of-Things domain.

Acknowledgements

The SSN ontology was developed through 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), CSIRO (Oznome project) and Marie Skłodowska-Curie Programme H2020-MSCA-IF-2014 (SMARTER project, under Grant No. 661180).

References


Appendix A. List of all SOSA/SSN Terms

Appendix B. SSN-System terms

Appendix C. Deprecated SSNX Terms
<table>
<thead>
<tr>
<th>SOSA/SSN term</th>
<th>replaces</th>
<th>SOSA/SSN term</th>
<th>replaces</th>
</tr>
</thead>
<tbody>
<tr>
<td>sosa:FeatureOfInterest</td>
<td>oldssn:FeatureOfInterest</td>
<td>ssn:hasProperty</td>
<td>oldssn:hasProperty</td>
</tr>
<tr>
<td>ssn:Property</td>
<td>oldssn:Property</td>
<td>ssn:isPropertyOf</td>
<td>oldssn:isPropertyOf</td>
</tr>
<tr>
<td>sosa:ObservableProperty</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>sosa:ActuatableProperty</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>sosa:Sample</td>
<td></td>
<td>sosa:hasSample</td>
<td>-</td>
</tr>
<tr>
<td>sosa:Platform</td>
<td>oldssn:Platform</td>
<td>sosa:hosts</td>
<td>oldssn:attachedSystem</td>
</tr>
<tr>
<td></td>
<td></td>
<td>sosa:isHostedBy</td>
<td>oldssn:onPlatform</td>
</tr>
<tr>
<td>sosa:Procedure</td>
<td>oldssn:Process</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ssn:hasInput</td>
<td>oldssn:hasInput</td>
<td>ssn:hasOutput</td>
<td>oldssn:hasOutput</td>
</tr>
<tr>
<td>ssn:Input</td>
<td>oldssn:Input</td>
<td>ssn:Output</td>
<td>oldssn:Output</td>
</tr>
<tr>
<td>ssn:implementedBy</td>
<td>oldssn:implementedBy</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ssn:implements</td>
<td>oldssn:implements</td>
<td></td>
<td></td>
</tr>
<tr>
<td>sosa:Sensor</td>
<td>oldssn:Sensor</td>
<td></td>
<td></td>
</tr>
<tr>
<td>sosa:observes</td>
<td>oldssn:observes</td>
<td></td>
<td></td>
</tr>
<tr>
<td>sosa:isObservedBy</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>sosa:Sampler</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>sosa:usedProcedure</td>
<td>oldssn:sensingMethodUsed</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>sosa:hasFeatureOfInterest</td>
<td>oldssn:featureOfInterest</td>
</tr>
<tr>
<td></td>
<td></td>
<td>sosa:isFeatureOfInterestOf</td>
<td>-</td>
</tr>
<tr>
<td>sosa:Observation</td>
<td>oldssn:Observation</td>
<td></td>
<td></td>
</tr>
<tr>
<td>sosa:madeObservation</td>
<td>oldssn:madeObservation</td>
<td></td>
<td></td>
</tr>
<tr>
<td>sosa:madeBySensor</td>
<td>oldssn:madeBySensor</td>
<td></td>
<td></td>
</tr>
<tr>
<td>sosa:observedProperty</td>
<td>oldssn:observedProperty</td>
<td></td>
<td></td>
</tr>
<tr>
<td>sosa:Sensing</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>sosa:madeSensing</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>sosa:madeBySensing</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ssn:Stimulus</td>
<td>oldssn:Stimulus, oldssn:SensorInput</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ssn:wasOriginatedBy</td>
<td></td>
<td>sns:detects</td>
<td>oldssn:detects</td>
</tr>
<tr>
<td></td>
<td></td>
<td>sns:isProxyFor</td>
<td>oldssn:isProxyFor</td>
</tr>
<tr>
<td>sosa:Result</td>
<td>oldssn:ObservationValue, oldssn:SensorOutput</td>
<td>sosa:hasSimpleResult</td>
<td>-</td>
</tr>
<tr>
<td>sosa:hasResult</td>
<td>oldssn:hasValue, oldssn:observationResult</td>
<td>sosa:hasResultingSample</td>
<td>-</td>
</tr>
<tr>
<td>sosa:isResultOf</td>
<td>oldssn:isProducedBy</td>
<td>sosa:isSamplingResultOf</td>
<td>-</td>
</tr>
<tr>
<td>sosa:resultTime</td>
<td>oldssn:observationResultTime</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>sosa:phenomenonTime</td>
<td>oldssn:observationSamplingTime</td>
</tr>
<tr>
<td>ssn:System</td>
<td>oldssn:System</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ssn:Deployment</td>
<td>oldssn:Deployment</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ssn:deployedSystem</td>
<td>oldssn:deployedSystem</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ssn:hasDeployment</td>
<td>oldssn:hasDeployment</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TOTAL SOSA: 13 classes, 21 object properties, 2 datatype properties</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TOTAL SSN: 6 classes, 15 object properties</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 2

Complete list of SOSA and SSN terms, and the terms in the old SSN that they supersede, if they exist, along with the formal relation between the new term and the old term. No relation means rdfs:seeAlso. The grouping of terms is only meant to ease readability.
<table>
<thead>
<tr>
<th>SOSA/SSN term</th>
<th>replaces</th>
<th>SOSA/SSN term</th>
<th>replaces</th>
</tr>
</thead>
<tbody>
<tr>
<td>ssn-system:qualityOfObservation</td>
<td>≡ oldssn:qualityOfObservation</td>
<td>ssn-system:condition</td>
<td>≡ oldssn:Condition</td>
</tr>
<tr>
<td>ssn-system:hasSystemCapability</td>
<td>⊒ oldssn:hasMeasurementCapability</td>
<td>ssn-system:hasSystemCapability</td>
<td>⊒ oldssn:hasMeasurementCapability</td>
</tr>
<tr>
<td>ssn-system:hasSystemProperty</td>
<td>⊒ oldssn:hasMeasurementProperty</td>
<td>ssn-system:hasSystemProperty</td>
<td>⊒ oldssn:hasMeasurementProperty</td>
</tr>
<tr>
<td>ssn-system:hasOperatingRange</td>
<td>≡ oldssn:hasOperatingRange</td>
<td>ssn-system:hasOperatingRange</td>
<td>≡ oldssn:hasOperatingRange</td>
</tr>
<tr>
<td>ssn-system:hasOperatingProperty</td>
<td>≡ oldssn:hasOperatingProperty</td>
<td>ssn-system:hasOperatingProperty</td>
<td>≡ oldssn:hasOperatingProperty</td>
</tr>
<tr>
<td>ssn-system:hasSurvivalRange</td>
<td>≡ oldssn:hasSurvivalRange</td>
<td>ssn-system:hasSurvivalRange</td>
<td>≡ oldssn:hasSurvivalRange</td>
</tr>
<tr>
<td>ssn-system:hasSurvivalProperty</td>
<td>≡ oldssn:hasSurvivalProperty</td>
<td>ssn-system:hasSurvivalProperty</td>
<td>≡ oldssn:hasSurvivalProperty</td>
</tr>
<tr>
<td>ssn-system:MaintenanceSchedule</td>
<td>≡ oldssn:MaintenanceSchedule</td>
<td>ssn-system:BatteryLifetime</td>
<td>≡ oldssn:BatteryLifetime</td>
</tr>
</tbody>
</table>

System properties for sensors


System properties for actuators

| ssn-system:ActuationRange           | -                          | ssn-system:ResponseTime               | ⊒ oldssn:ResponseTime    |

System properties for any system

| ssn-system:Accuracy                 | ⊒ oldssn:Accuracy          | ssn-system:Drift                       | ⊒ oldssn:Drift           |
| ssn-system:Frequency                | ⊒ oldssn:Frequency         | ssn-system:Latency                     | ⊒ oldssn:Latency         |
| ssn-system:Precision                | ≡ oldssn:Precision         | ssn-system:Resolution                  | ⊒ oldssn:Resolution      |
| ssn-system:Resolution               | -                            | ssn-system:ResponseTime                | ⊒ oldssn:ResponseTime    |

Operating ranges for any system

| ssn-system:hasOperatingRange        | ≡ oldssn:hasOperatingRange  | ssn-system:hasOperatingRange           | ≡ oldssn:hasOperatingRange |
| ssn-system:hasOperatingProperty     | ≡ oldssn:hasOperatingProperty| ssn-system:hasOperatingProperty        | ≡ oldssn:hasOperatingProperty |

Survival ranges for any system

| ssn-system:hasSurvivalRange         | ≡ oldssn:hasSurvivalRange  | ssn-system:hasSurvivalRange            | ≡ oldssn:hasSurvivalRange |
| ssn-system:hasSurvivalProperty      | ≡ oldssn:hasSurvivalProperty| ssn-system:hasSurvivalProperty         | ≡ oldssn:hasSurvivalProperty |

TOTAL SSN-System: 23 classes, 8 object properties

Table 3

Complete list of terms in SSN-System, and the terms in the old SSN that they supersede, if they exist, along with the formal relation between the new term and the old term. The grouping of terms is only meant to ease readability.
### Table 4

<table>
<thead>
<tr>
<th>Term</th>
<th>Cause</th>
<th>Alignment to concepts in Sosa/SSN</th>
</tr>
</thead>
<tbody>
<tr>
<td>oldssn:startTime</td>
<td>Unused</td>
<td>-</td>
</tr>
<tr>
<td>oldssn:endTime</td>
<td>Unused</td>
<td>-</td>
</tr>
<tr>
<td>oldssn:Device</td>
<td>Misused</td>
<td>⊑ sosa:Platform ⊓ sosa:System</td>
</tr>
<tr>
<td>oldssn:SensingDevice</td>
<td>Misused</td>
<td>⊑ sosa:Platform ⊓ sosa:Sensor</td>
</tr>
<tr>
<td>oldssn:Sensing</td>
<td>Unused</td>
<td>⊑ sosa:Procedure</td>
</tr>
<tr>
<td>oldssn:DeploymentRelatedProcess</td>
<td>Unused</td>
<td>-</td>
</tr>
<tr>
<td>oldssn:deploymentProcessPart</td>
<td>Unused</td>
<td>-</td>
</tr>
<tr>
<td>oldssn:ofFeature</td>
<td>Unused</td>
<td></td>
</tr>
<tr>
<td>oldssn:SensorDataSheet</td>
<td>Unused</td>
<td></td>
</tr>
</tbody>
</table>

**TOTAL:** 9 deprecated terms

Terms in the old SSN that have no equivalent in the Sosa or SSN, and are simply deprecated. *Unused* terms were deprecated because no implementation evidence was found, and no requirement justified their inclusion in SSN. *Misused* terms were proven to be prone to misinterpretation and were deprecated in SSN.