Paper submitted to the 9th International Semantic Sensor Networks Workshop – SSN 2018.
See https://ssn2018.github.io/ for the reviews, if available,
and a link to the final version if this contribution was accepted.

VSSo: A Vehicle Signal and Attribute Ontology

Benjamin Klotz[1,2] [0000−0002−9008−2120], Raphaël Troncy[1] [0000−0003−0457−1436], Daniel Wilms[2] [0000−0002−0451−0265], and Christian Bonnet[1] [0000−0002−3733−7227]

[1] EURECOM, Sophia Antipolis, France
firstname.lastname@eurecom.fr
[2] BMW Research, New Technologies, Innovation, Munich, Germany

Abstract. Application developers in the automotive domain have to deal with thousands of different signals, represented in highly heterogeneous formats, and coming from various car architectures. This situation prevents the development and connectivity of modern applications. We hypothesize that a formal model of car signals, in which the definition of signals are uncorrelated with the physical implementations producing them, would improve interoperability. In this paper, we propose VSSo, a car signal ontology that derives from the automotive standard VSS, and that follows the SSN/SOSA pattern for representing observations and actuations. This ontology is comprehensive while being extensible for OEMs, so that they can use additional private signals in an interoperable way. We developed a simulator for interacting with data modeled under the VSSo ontology pattern available at http://automotive.eurecom.fr/simulator/query

Keywords: ontology, automotive, signal, sensor, VSS

1 Introduction

Current and future automotive applications rely on the ability to manage highly heterogeneous data, coming from cars themselves or from other parties such as web services, or connected things like smart homes and smart cities. In this context, vehicle data needs to be interoperable in order to be handled by remote applications and services regardless of the brand, model, and internal network architecture of each connected vehicle. This is actually challenging today as a developer needs deep insights into the architecture of a vehicle[3] in order to have access and to process data coming from a vehicle signal. In addition, information about signal metadata is needed in order to interpret the returned values. As soon as the internal architecture changes, the developer has to update the implementation and will need the same prior knowledge. This might be the case already with different models of the same brand.

[3] http://www.ieee802.org/1/files/public/docs2013/new-tsn-diarra-osi-layers-in-automotive-networks-0313-v01.pdf

Paper submitted to the 9th International Semantic Sensor Networks Workshop – SSN 2018.
See https://ssn2018.github.io/ for the reviews, if available,
and a link to the final version if this contribution was accepted.

There is a trend of publishing understandable Web services[4] and APIs by OEMs[5] (Original Equipment Manufacturers), that enable the access to specific car signals and use tree structures to represent car data. This structure was originally specified by the GENIVI Alliance and W3C under the name VSS (Vehicle Signal Specification)[6]. This specification states, for example, that the speed measured by the GPS can be accessed by going through a tree from the *Cabin*, the *Infotainment* and then the *GPS* branches to finally reach the *Speed* signal. Hence, car signals data is accessed within some context (the branch of the vehicle that generates it). VSS is the building block of the VISS [7] (Vehicle Information Service Specification) and VIAS [3] (Vehicle Information API Specification) specifications which are under development within the W3C Automotive Working Group. However, this structure does not solve entirely the issue of interoperability. Indeed, with a huge amount of sensors embedded in most modern cars, many of them are obscure to non automotive experts and rely on non standard units. Therefore, the knowledge of how to interpret values should also be represented. A single API for all vehicles, for instance, would fail as soon as the unit system changes. We propose to use semantic technologies for addressing the challenge of defining a formal model of car signals [4].

Many ontologies have been developed in order to solve problems in the automotive domain. In 2003, [17] proposed an ontology-based data access for car. [19] describes the relationship between components, failures and their symptoms. [6] proposes an automotive ontology describing the user's actions and car context. More generally, several research projects proposed ontology-based representation of some car context to provide advanced driver-assistance systems (ADAS) [24, 2, 16, 22, 11], but they are not complete or extensible, nor they are automotive standards.

W3C and OGC have developed standards for defining systems with their signals. The Semantic Sensor Network[7] (SSN) ontology [10] is an ontology for describing sensors and their observations, the involved procedures, the studied features of interest, the samples used to do so and the observed properties, as well as actuators and actuations. SSN follows a horizontal and vertical modularization architecture by including a lightweight but self-contained core ontology called SOSA[8] [8] (Sensor, Observation, Sample, and Actuator) for its elementary classes and properties, that was released in October 2017. Both SSN and SOSA are domain independent. There are applications built using them in different domains including satellite imagery, large-scale scientific monitoring, industrial and household infrastructures, social sensing, citizen science, observation-driven ontology engineering, and the Web of Things (WoT) [10].

[4] `https://ifttt.com/bmwlabs`
[5] `https://developer.mercedes-benz.com/`, `http://www.porsche-next-oi-competition.com/`, `https://developer.psa-peugeot-citroen.com/inc/`
[6] `https://github.com/GENIVI/vehicle_signal_specification`
[7] `http://www.w3.org/ns/ssn/`
[8] `http://www.w3.org/ns/sosa/`

Paper submitted to the 9th International Semantic Sensor Networks Workshop – SSN 2018.
See https://ssn2018.github.io/ for the reviews, if available,
and a link to the final version if this contribution was accepted.

WoT is one mean of abstracting connected things such as car. It is also, per se, highly dependent on its data model to enable interoperability between Web Things. As the IoT (Internet of Things) is highly heterogeneous, there are too many standards and protocols in competition at low layers of the OSI model[9]. The WoT (Web of Things) focuses on the application layer using Web technologies and is agnostic to the transport and physical layers[10]. We enriched the generic WoT approach by adding domain semantics based on SSN/SOSA patterns for interacting with a connected car [13].

`schema.org` is also a *de facto* standard for marking up web pages with structured metadata to facilitate Web search. There are extensions dedicated to the IoT[11] and the automotive domain[12]. They are still under development, notably for aligning some WoT concepts with SSN/SOSA, and currently only define a small set of static properties describing cars. This last extension comes from the work of the W3C Automotive Ontology Working Group[13] which started with the goal of describing cars in e-commerce, based on the some ontologies describing vehicles such as the Vehicle Sales Ontology[14] (VSO), the Used Cars Ontology[15], the Car Option Ontology[16] and the Volkswagen Vehicle Ontology[17]. Those models define properties for describing static information of vehicles that are useful for marking up cars to sell[18]. However, they do not enable developers to describe the sensors embedded in cars, nor their signals.

We therefore observe a gap between the need for data interoperability and the current state of the art in terms of car modeling. We see a need for an ontology focusing on car signals and sensors. We also identify another requirement: such an ontology should be compliant with automotive standard such as VSS or ISO 20078 [1] or follow best modeling practices in order to be used. We require such an ontology to be comprehensive enough to cover most known signals while being extensible by OEMs. The remainder of this paper is structured as follow. First, we list our requirements in Section 2. We describe how we have converted the VSS automotive standard into an ontology in Section 3. We show how this ontology can be used and consumed in Section 4 before concluding and describing future work in Section 5.

[9] `https://www.postscapes.com/internet-of-things-protocols/`
[10] `https://webofthings.org/2016/01/23/wot-vs-iot-12/`
[11] `iot.schema.org`
[12] `auto.schema.org`
[13] `http://www.automotive-ontology.org/`
[14] `http://www.heppnetz.de/ontologies/vso/ns`
[15] `http://ontologies.makolab.com/uco/ns.html`
[16] `http://semanticweb.org/wiki/Car_Options_Ontology.html`
[17] `http://www.volkswagen.co.uk/vocabularies/vvo/ns`
[18] `https://www.w3.org/community/gao/2017/09/12/toyota-motor-europe-use-of-schema-org-and-auto-schema-org-vocabularies/`

Paper submitted to the 9th International Semantic Sensor Networks Workshop – SSN 2018.
See https://ssn2018.github.io/ for the reviews, if available,
and a link to the final version if this contribution was accepted.

2 Requirements

A good car signal ontology should enable a web developer to query and extract knowledge from a car signal database with no deep expertise in the automotive domain. In this section, we define a set of competency questions, which we will later use as a mean of evaluating the produced ontology.

2.1 Description of car attributes

There is a need to define a number of static properties or attributes describing either a complete vehicle or its parts (later named branches) - such as the engine - and their position.

- What are the attributes of a car and what do they express?
- How many attributes does a car have?
- What is the model of this car?
- What is the brand of this car?
- What is the VIN of this car?
- When was produced this car?
- What are the dimensions of this car?
- What type of fuel does this car need?
- What type of transmission does this car have?
- What are the characteristics of this engine?
- How many doors does this car have?
- How many seats does this car have?
- On which side is located the steering wheel in this car?

2.2 Description of car signals

A car contains numerous sensors that produce signals. We provide competency questions related to those signals, that are linked to branches of a vehicle in some position and that generate values in some unit systems.

- Is there a signal measuring the steering wheel angle?
- Which signals are controllable?
- Which signals are both observable and actuable?
- How many sensors does this car contain?
- How many different speedometers does this car contain?
- In which part of this car is produced the signal `vss:LongitudinalAcceleration`?
- Which signals measure a temperature and in which part are they located in the car?
- What unit type does the signal `vss:VehicleYaw` use?
- What are the characteristics of the sensor producing the signal “Travelled-Distance” in the OBD branch?
- What are the maximum values allowed for all signals from a Vehicle?

Paper submitted to the 9th International Semantic Sensor Networks Workshop – SSN 2018.
See https://ssn2018.github.io/ for the reviews, if available,
and a link to the final version if this contribution was accepted.

2.3 Description of dynamic car states

When being used, car sensors will generate a lot of values that depend on time and space. One should be able to query the current values of the signals as well as past historical ones. This leads to additional competency questions.

- What is the current gear?
- What are the values of all signals representing the speed of this car in this moment?
- Which windows are currently open?
- What is the local temperature on the driver side now?
- What are the current values of signals defining the driver seat position?
- When was the last time the speed was over 100 km/h?
- When and where was the last time the driver's door was unlocked?
- What was the maximal speed reached by the car?

We hypothesis that the generic modeling patterns defined in the SSN/SOSA ontology [8] is adequate to describe observations and that an additional vocabulary is needed to define the specific terms in the automotive domain.

3 Development of VSSo

We developed VSSo, a vehicle signal ontology based on the GENIVI and W3C standard data model VSS (Vehicle Signal Specification). The ontology is available at `https://github.com/klotzbenjamin/vss-ontology/`.

3.1 VSS

The Vehicle Signal Specification defines a tree containing 451 *Branches*, 43 *Attributes* and 1060 *Signals* that aim to represent car data (Fig. 1). The specification states that:

- *Branches* are car parts or components. They are represented as nodes in the VSS tree. Branches can contain other branches or signals and attributes. For instance the top branches in the VSS tree are *Body*, *ADAS*, *Cabin*, *Chassis*, *Drivetrain*, *OBD* and *Vehicle*.
- *Attributes* are the static information about a car that should not change over time and space. Attributes are represented as leaves in the VSS tree. For instance, the dimensions and VIN (Vehicle Identification Number) of a car are attributes of the *Vehicle* branch. Attributes are defined by a path starting with "Attribute" and defining its position in the VSS tree. For instance the VIN is `Attribute.Vehicle.VehicleIdentification.VIN`. They also have entries such as a description, a type, a unit or restrictions on values. All properties defined in `http://auto.schema.org` are attributes in VSS.

– *Signals* are the dynamic information about a car that is either produced by a sensor, consumed by an actuator or properties of complex embedded systems. Signals are also represented as leaves in the VSS tree. For instance, `Signal.Drivetrain.Transmission.Speed` is the car speed, measured in the *Transmission* branch. Signals, like attributes, have entries providing a description, a type, and potentially a unit and restrictions on values.

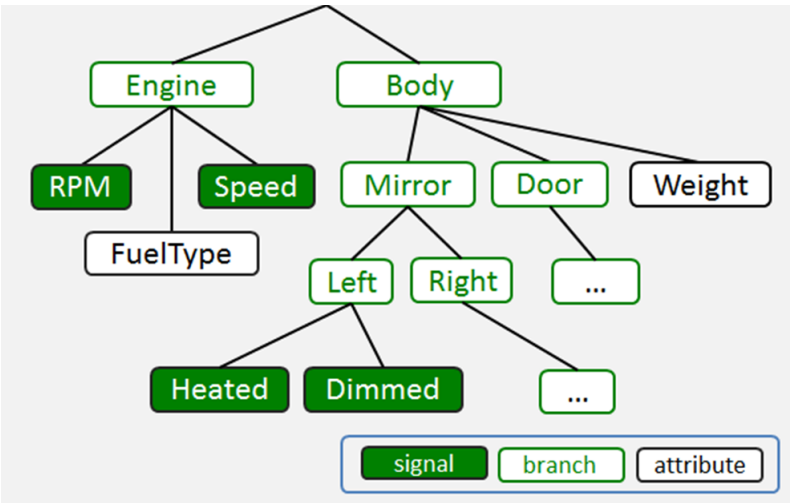

Fig. 1. The GENIVI Vehicle Signal Specification structure

In its original form, VSS did not contain information about sensors or actuators producing or consuming data. In order to describe the difference between signals measuring the same phenomenon, but sensed by different sensors, such as the car speed, we added new entries in VSS signals. We also corrected some entries to make VSS more consistent, especially in the naming convention and choice of standard units. Those corrections have been approved by GENIVI and are now part of evolution of this standard.

VSS is meant to be a technology-independent specification for car data. This means that a component or signals specific to a particular brand or car model should not define a specific technology as other competing ones exist to do the same task. For instance, the traveled distance is measured by an *Odometer* regardless of the technology used.

3.2 General modeling pattern

The general idea behind the design of the VSS ontology is to take advantage of the structure of VSS. All branches are part of a complete tree, as sub-branches

Paper submitted to the 9th International Semantic Sensor Networks Workshop – SSN 2018.
See https://ssn2018.github.io/ for the reviews, if available,
and a link to the final version if this contribution was accepted.

of bigger branches. This structure gives a more understandable meaning to signals. Therefore, we reuse it in a component-based pattern using subclasses of `vss:Branch` linked with the transitive object property `vss:partOf`. This means that a VSS *Branch* is used to generate a new class, and the mother or children branches are attached to it with a `vss:partOf` property (Listing 1.1).

Listing 1.1. `vss:Drivetrain` is an example of a generated class part of `vss:Vehicle`

```
vss:Drivetrain a rdfs:Class, owl:Class;
    rdfs:subClassOf vss:Branch;
    rdfs:subClassOf [
        a owl:Restriction;
        owl:onProperty vss:partOf;
        owl:allValuesFrom vss:Vehicle
    ];
    rdfs:label "Drivetrain"@en;
    rdfs:comment "Drivetrain. All body components"@en.
```

The second interesting structural aspect of VSS is the set of entries defining VSS concepts. Indeed, attributes, branches and signals are all defined by at least a name, a type and a description. These entries allow the generation of one class per VSS concept, with a RDFS label and comment. Attributes and signals also have additional entries, such as a unit, or a set of potential values (sometimes a minimum and maximum values) and a sensor or actuator. All these entries define the specific details of an attribute or signal, and make more sense to a machine than a label or description (Listing 1.2).

Listing 1.2. `vss:AmbientAirTemperature` is a signal measured by a `vss:Thermometer` in `qudt:DegreeCelcius`

```
vss:AmbientAirTemperature a rdfs:Class, owl:Class;
    rdfs:subClassOf vss:ObservableSignal;
    rdfs:label "AmbientAirTemperature"@en;
    rdfs:comment "Signal.Vehicle.AmbientAirTemperature :
        Ambient air temperature"@en;
    rdfs:subClassOf [
        a owl:Restriction;
        owl:onProperty sosa:isObservedBy;
        owl:allValuesFrom vss:Thermometer
    ];
    rdfs:subClassOf [
        a owl:Restriction;
        owl:onProperty qudt:unit;
        owl:allValuesFrom qudt:TemperatureUnit
    ].
```

We generate a datatype property for each VSS attribute which are subproperties of a generic `vss:attribute` datatype property. All those attributes being static since the values do not evolve in time and space, there is no need to

model them using a pattern involving dynamic observations. VSS attributes are attached to VSS branches which is materialized in the domain of those properties, while their range makes use of a custom datatype (Listing 1.3).

Listing 1.3. `vss:tankCapacity` is an attribute of the `vss:FuelSystem` branch

```
vss:tankCapacity a owl:DatatypeProperty;
    rdfs:subPropertyOf vss:attribute;
    rdfs:label "TankCapacity"@en;
    rdfs:comment "Attribute.Drivetrain.FuelSystem.Tank-
        Capacity:
        Capacity of the fuel tank in liters"@en;
    rdfs:domain vss:FuelSystem;
    rdfs:range cdt:volume.
```

Signals, however, are going to be observed over time and space and there is a need for an adapted modeling pattern taking dynamics into account. In order to model it, we take advantage of the SSN/SOSA pattern for modeling sensors, actuators, observable and actuatable properties, observations and actuations. SOSA uses the triplets (*Observation, ObservableProperty, Sensor*) and (*Actuation, ActuatableProperty, Actuator*) where the first element defines the abstraction data, the second the signal and the third the appliance producing or consuming the data. Observations and Actuations contextualize the data with properties such as `sosa:FeatureOfInterest` (e.g. a car), the `sosa:Result` or `sosa:SimpleResult` depending on how units are defined, as well as `sosa:phenomenonTime` and `geo:lat`, `geo:long` for the spatiotemporal context of the observation or actuation (Fig. 2).

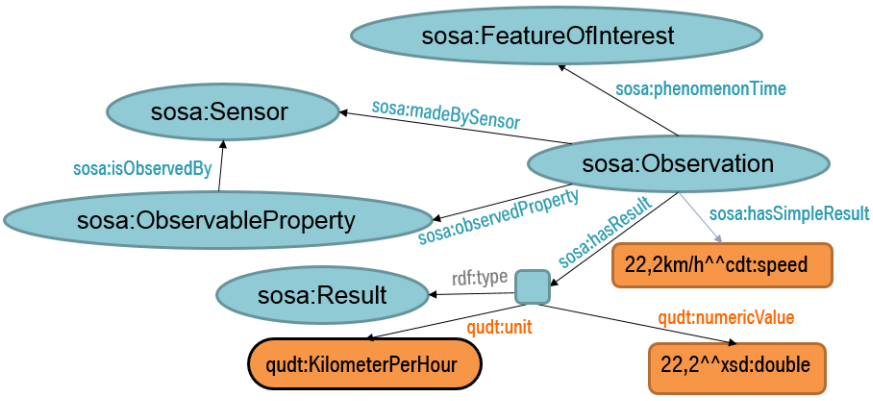

Fig. 2. The SOSA modeling pattern for sensors and observable properties

Paper submitted to the 9th International Semantic Sensor Networks Workshop – SSN 2018.
See https://ssn2018.github.io/ for the reviews, if available,
and a link to the final version if this contribution was accepted.

SSN/SOSA does not define a unique unit ontology, but it is open to use multiple ones. The examples provided in the specification[19] use QUDT[20][9], OM[21][22] [20] or a custom datatype[23] [15]. The main unit ontologies have been compared in [12]: OM is the largest one with relatively few issues, while QUDT is a medium sized ontology with some inferential inconsistencies but a partial mapping with schema.org[24]. The authors of [15] have developed a custom datatype supporting the units from the UCUM (Unified Code for Units of Measure) system[25]. In order to remain open, we only set restrictions on unit systems in QUDT and let the user choose the units freely.

3.3 Modeling problems and new VSS policies

Several exceptions and issues prevent the trivial generation of a healthy ontology from VSS. Some concepts share the same name or require clarification, signals must be compliant with the SOSA pattern and there are branches defining position concepts that are not relevant for a VSS ontology.

Clarification of concept names

Homonymy. VSS relies on a full path to define an attribute or a signal. Usually, the path contains all the context to make sense with a generic name. For instance `Signal.Drivetrain.Engine.Speed`, is clearly the rotation speed of the engine while `Signal.Cabin.Infotainment.Navigation.CurrentLocation.Speed` is the vehicle speed measured by the GPS. However they would both generate a class `vss:Speed` if we would take the leaf of the tree as a basis. Therefore, VSS concepts are renamed for clarification. In the same example, they will generate `vss:EngineSpeed` and `vss:VehicleSpeed`.

Synonymy. Sometimes, two different path in the VSS tree actually refer to the same concept. This happens when the same phenomenon can be measured by more than one sensor or in different parts of the vehicle. For example, the signals `Signal.Drivetrain.Transmission.Speed` and `Signal.Cabin.Infotainment.Navigation.CurrentLocation.Speed` both measure the speed of the car, one in the gearbox using rotation speed measures and the other one using GPS coordinates. The class `vss:VehicleSpeed` is here unique. Its instances differ given the sensors that will produce the value and the branch hosting the sensor (Fig. 3).

[19] `https://www.w3.org/TR/vocab-ssn/#examples`
[20] `http://www.qudt.org`
[21] Version 1.8.6 from `http://www.wurvoc.org/vocabularies/om-1.8/`
[22] Version 2.0.6 from `https://github.com/HajoRijgersberg/OM`
[23] `https://ci.mines-stetienne.fr/lindt/v2/custom_datatypes.html`
[24] `https://www.w3.org/TR/2016/NOTE-tabular-data-primer-20160225/#units-of-measure`
[25] `http://unitsofmeasure.org/ucum.html`

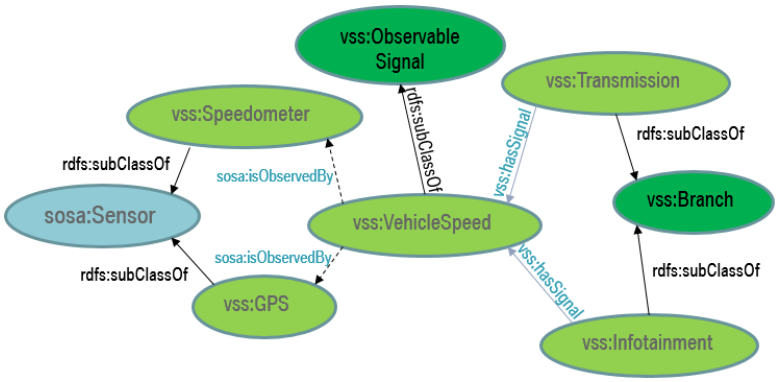

Fig. 3. Two signals representing the same concept: the vehicle speed

Restriction required by the SOSA pattern In SOSA, there is no definition of specific signals but only `sosa:ObservableProperty` and `sosa:ActuatableProperty`. These classes are not mutually exclusive and simply define signals that are meant to be read and signals meant to be written.

Signals are observable, actuatable or both. We define two main signal classes in the VSS ontology: `vss:ObservableSignal`, as a subclass of `sosa:ObservableProperty`, and `vss:ActuatableSignal` subclass of `sosa:ActuatableProperty`. All signals in VSS are subclasses of at least one of them. For instance, `vss:VehicleSpeed` is only measured and is therefore a subclass of `sosa:ObservableProperty`, while `vss:MirrorHeating` only acts on the mirror and is a subclass of `sosa:ActuatableProperty`. Many signals are subclasses of both. The choice of making a signal observable or actuatable is based on the existence of the sensor and actuator entries of each VSS Signal. If it has a sensor, it is observable, but if it has an actuator, it is actuatable.

All signals have at least a sensor or actuator. In order to be compliant with SOSA, we must define a `sosa:Sensor` for all `sosa:ObservableProperty` and a `sosa:Actuator` for all `sosa:ActuatableProperty`. This means that the entries we added in VSS to defines those devices are used to create classes, subclasses of either `sosa:Sensor` or `sosa:Actuator`. These sensors and actuators should be as technology-independent as possible, as their physical instances vary from one OEM to another. Some signals relate to complex systems such as the infotainment system where there are no physical sensors or actuators. In this case, a virtual system defines the sensor/actuator producing or consuming the data without being a physical device.

Branch structure modeling choices The VSS tree structure contains choices that prevents an automatic generation of RDF classes for branches.

Paper submitted to the 9th International Semantic Sensor Networks Workshop – SSN 2018.
See https://ssn2018.github.io/ for the reviews, if available,
and a link to the final version if this contribution was accepted.

All branches are `vss:partOf vss:Vehicle`. The path defining attributes and signals begins with the top element of the tree, being either "Attribute" or "Signal". The modeling choice would require the top branch to be the complete vehicle that contains all branches. There is, nevertheless, a branch among the top one called "Vehicle" containing attributes and signals about the full vehicle, such as its VIN. We take this branch as the top one containing all other branches (Fig. 4).

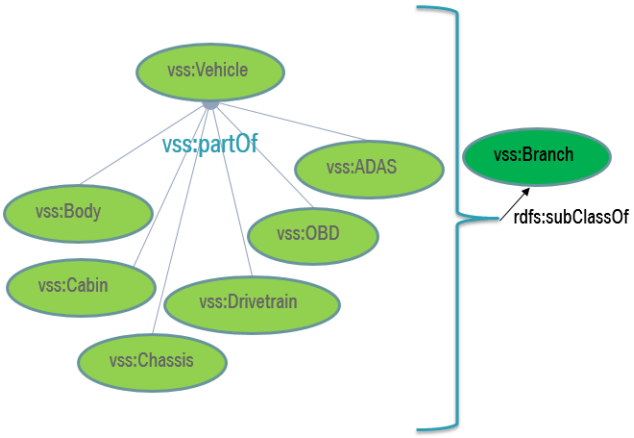

Fig. 4. The vss:Vehicle branch is taken as the one containing all other and the full vehicle

Position-related concepts are not branches In VSS, the path to certain attributes and signals contains the position of certain branches. This is especially the case for elements that exist multiple times within one car, such as doors, seats and mirrors. For instance, there are signals `Door.Left.IsLocked` and `Mirror.Right.Tilt`. It is not desired to have classes defining "left" and "right". A solution could be to define a class per signal in VSS, but the result would not be consistent with the goal of having generic signal classes. Instead, we decide to model the hosting branches with a property `vss:position`. This defines instances of such branches with the correct positions and still refer to a unique class. Using the same example, a door instance would have `vss:position` "Left"@en and the mirror instance a `vss:position` "Right"@en.

Paper submitted to the 9th International Semantic Sensor Networks Workshop – SSN 2018.
See https://ssn2018.github.io/ for the reviews, if available,
and a link to the final version if this contribution was accepted.

3.4 Evaluation

In order to evaluate the coverage of the VSS ontology, we tried to write SPARQL queries for all competency questions described in the Section 2[26]. We generate synthetic traces data using the VSSo ontology.

What are the attributes of a car and what do they express? This query retrieves the static information about a car: its attributes.

```
SELECT ?branch ?attribute ?value
WHERE {
  ?attribute  rdfs:subPropertyOf  vss:attribute.
  ?branch ?attribute ?value. }
```

What are the attributes of the chassis? This query is interesting for focusing on only one branch of the car.

```
SELECT ?attribute ? value
WHERE {
  ?attribute  rdfs:subPropertyOf  vss:attribute.
  ?branch ?attribute ?value;
            a vss:Chassis. }
```

Which unit system does the signal `vss:VehicleYaw` *use?* The ontology enables to perform queries on units.

```
SELECT ?unitsystem
WHERE {
  ?yaw  a vss:VehicleYaw;
        qudt:unit ?unitsystem. }
```

What is the current gear? A developer should only be required to know the URI of a signal to retrieve its last value and metadata[27]. In this example, with the SSN/SOSA observations, we check that the current time is the time of the observation and retrieve the value and unit.

```
SELECT ?signal ?result ?time
WHERE {
  ?signal a vss:CurrentGear.
  ?obs a sosa:Observation;
          sosa:observedProperty ?signal;
          sosa:hasSimpleResult ?result;
```

[26] A more complete list is available at `https://github.com/klotzbenjamin/vss-ontology`

[27] In the case of time-related query, we assume we can define the current time with a function NOW().

Paper submitted to the 9th International Semantic Sensor Networks Workshop – SSN 2018.
See https://ssn2018.github.io/ for the reviews, if available,
and a link to the final version if this contribution was accepted.


```
            sosa:phenomenonTime ?time.
  FILTER(?time == NOW())
}
```

Which windows are currently open? In this case, we consider the position of a car component, to make sure that one can define it in instances of car branches and signals. The window position is in percent, so if a signal is observed with a value different from 100, the branch that contains it is kept and we look at the property `vss:position` of the remaining branches.

```
SELECT ?position
WHERE {
  ?windowPosition a vss:WindowPosition.
  ?window vss:hasSignal ?windowPosition.
  ?obs a sosa:Observation;
          sosa:observedProperty ?windowPosition;
          sosa:hasResult ?result;
          sosa:PhenomenonTime ?time.
  FILTER(?time == NOW())
  ?result qudt:numericValue ?value.
  FILTER(?value < 100)
  ?window vss:position ?position.
}
```

VSSo fits our requirements of being based on an automotive standard and semantically enriching car data. Furthermore, with more than 300 different signals and 50 attributes, VSSo defines more concepts than all ontologies, vocabularies and schemata from the state of the art, making it more complete. Finally, because VSSo is based on a specification meant to be extended, it is also easy to extend, as we will see in Section 4.2.

4 Usage

The most general use case for VSSo is the creation and query of triples about observations. Such triples are created following the SOSA pattern. For instance, an observation of a temperature in degrees Celsius would be written as in Listing 1.4, with description of the geolocation with `geo:lat` and `geo:long`.

Listing 1.4. An Observation of a temperature

```
:AmbientAirTemperature/observation171 a sosa:Observation ;
    geo:lat "48.151099"^^xsd:long ;
    geo:long "11.540354"^^xsd:long ;
    sosa:hasFeatureOfInterest :MyCar ;
    sosa:hasResult [ a qudt-1-1:QuantityValue ;
            qudt-1-1:numericValue "-0.5" ;
            qudt-1-1:unit qudt-unit-1-1:DegreeCelcius] ;
```

```
sosa:madeBySensor :MyThermometer ;
sosa:observedProperty :MyAmbientAirTemperature ;
sosa:phenomenonTime "2018-01-22T08:17:15.67Z"
^^xsd:dateTime.

:MyThermometer a vss:Thermometer.
:MyAmbientAirTemperature a vss:AmbientAirTemperature.
:MyCar a vss:Vehicle.
```

4.1 Use Cases benefiting from VSSo

We used VSSo in various use cases to highlight its benefits.

On the fly generation of triples We developed a simulator for car data, producing RDF triples following the SSN/SOSA and VSSo ontologies. These observations are available for trying the queries presented in Section 3.4. A public sparql endpoint is available at `automotive.eurecom.fr/simulator/query`.

Generation and Annotation of Semantic Trajectories A current challenge in trajectory pattern mining [23] is the production and analysis of car data. Among the benefits of such an analysis is the knowledge about patterns but also the understanding of trajectories for drivers [21], outlier detection [5], and predictions [18]. The current trend is to use smartphones and a limited set of signals, mostly time and location. There has been some research on the case of the automotive domain [21, 5], but it is mostly limited to open datasets of fleets of taxis or from one unique vehicle.

VSSo makes it easier to combine data from a heterogenous fleet [14]. We recorded data from a BMW car and developed a interfacing server[28] to interact with these traces using the VSSo model. In this demonstration, we create a static graph decribing the car's attributes, then fill it with `sosa:Observation` and use a simple reasoner to label trajectories segments between consecutive observations.

A car as a Web Thing In another demonstration, we used VSSo as a domain ontology to access a connected car as a Web Thing as part of the W3C Web of Things Working Group experiments [13]). We created Thing Description[29] files containing interactions, enriched with VSSo concepts about the signals they are interacting with.

[28] `automotive.eurecom.fr/trajectory`

[29] `https://github.com/w3c/wot/tree/master/plugfest/2018-prague/TDs/EURECOM-TD`

Paper submitted to the 9th International Semantic Sensor Networks Workshop – SSN 2018.
See https://ssn2018.github.io/ for the reviews, if available,
and a link to the final version if this contribution was accepted.

4.2 VSSo Extensions

Just like VSS is meant to be extended with private signals and branches, VSSo can import new concepts defined in other namespaces. In order to do so, a developer can directly use VSSo and its patterns to manually create new attributes, branches and signals. Another solution consists in writing the VSS extension in vspec format, and generate a new ontology. However this second solution requires a step of validation afterwards. We extended the generator with a health check script[30] in order to reduce the effort of manual validation.

For instance, an OEM can define a private signal for a new embedded camera. In order to use it, a developer will define this camera as part of the VSSo extension in a new namespace.

5 Conclusion and Future Work

In this paper, we identified a gap in formal definition of car signals and sensors. We used some best practices both from the Semantic Web community and the automotive standards to propose VSSo, an ontology developed on top of the SSN/SOSA recommendation. This new formal representation of car signals and attributes allows semantic queries. Among the fields that can benefit from it are fleet monitoring and trajectory mining, contextual representation of a car and interaction between any car and web services.

References

1. I. 20078. Road vehicles – Extended vehicle (ExVe) 'web services'. Standard, ISO, 2018.
2. A. Armand, D. Filliat, and J. Ibaez-Guzman. Ontology-based context awareness for driving assistance systems. In *IEEE Intelligent Vehicles Symposium Proceedings*, pages 227–233, 2014.
3. M. Aro, S. Urata, and P. Kinney. Vehicle Information API Specification. Working draft, W3C, 2017.
4. P. Barnaghi, W. Wang, C. Henson, and K. Taylor. Semantics for the Internet of Things: early progress and back to the future. *International Journal on Semantic Web and Information Systems (IJSWIS)*, 8(1):1–21, 2012.
5. A. R. de Aquino, L. O. Alvares, C. Renso, and V. Bogorny. Towards Semantic Trajectory Outlier Detection. In *14th GeoInfo Conference (GEOINFO)*, pages 115–126, 2013.
6. M. Feld and C. Müller. The Automotive Ontology: Managing Knowledge Inside the Vehicle and Sharing it Between Cars. In *3rd International Conference on Automotive User Interfaces and Interactive Vehicular Applications*, pages 79–86, Salzburg, Austria, 2011.
7. K. Gavigan, A. Crofts, P. Kinney, and W. Lee. Vehicle Information Service Specification. Candidate recommendation, W3C, 2018.

[30] `https://github.com/klotzbenjamin/vss-ontology/tree/master/rdf-generation`

8. A. Haller, K. Janowicz, S. Cox, M. Lefranois, K. Taylor, D. L. Phuoc, J. Lieberman, R. Garca-Castro, R. Atkinson, and C. Stadler. The Modular SSN Ontology: A Joint W3C and OGC Standard Specifying the Semantics of Sensors, Observations, Sampling, and Actuation. under review, 2018.
9. R. Hodgson, P. J. Keller, J. Hodges, and J. Spivak. QUDT-quantities, units, dimensions and data types ontologies. *USA, Available from: http://qudt. org [March 2014]*, 2014.
10. K. Janowicz, S. Cox, K. Taylor, D. L. Phuoc, M. Lefrançois, and A. Haller. Semantic Sensor Network Ontology. Recommendation, W3C, 2017.
11. S. Kannan, A. Thangavelu, and R. Kalivaradhan. An Intelligent Driver Assistance System (I-DAS) for Vehicle Safety Modelling using Ontology Approach. *International Journal Of UbiComp (IJU)*, 1(3), 2010.
12. J. M. Keil and S. Schindler. Comparison and Evaluation of Ontologies for Units of Measurement. *Semantic Web journal*, 2018.
13. B. Klotz, S. K. Datta, D. Wilms, R. Troncy, and C. Bonnet. A Car as a Semantic Web Thing: Motivation and Demonstration. In *2nd Global Internet of Things Summit (GIoTS)*, Bilbao, Spain, 2018.
14. B. Klotz, R. Troncy, D. Wilms, and C. Bonnet. Generating Semantic Trajectories Using a Car Signal Ontology. In *The Web Conference (WWW)*, Lyon, France, 2018.
15. M. Lefrançois and A. Zimmermann. Supporting arbitrary custom datatypes in RDF and SPARQL. In *International Semantic Web Conference (ISWC)*, pages 371–386, 2016.
16. M. Madkour and A. Maach. Ontology-based context modeling for vehicle context-aware services. *Journal of Theoretical and Applied Information Technology*, 34(2), 2011.
17. A. Maier, H.-P. Schnurr, and Y. Sure. Ontology-Based Information Integration in the Automotive Industry. In *2nd International Semantic Web Conference (ISWC)*, pages 897–912, 2003.
18. A. Monreale, F. Pinelli, R. Trasarti, and F. Giannotti. WhereNext: A Location Predictor on Trajectory Pattern Mining. In *15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, pages 637–646, Paris, France, 2009.
19. A. Reymonet, J. Thomas, and N. Aussenac-gilles. Ontology Based Information Retrieval: an application to automotive diagnosis. In *20th International Workshop on Principles of Diagnosis (DX)*, pages 9–14, Stockholm, Sweden, 2009.
20. H. Rijgersberg, M. Van Assem, and J. Top. Ontology of Units of Measure and Related Concepts. *Semantic Web journal*, 4(1):3–13, 2013.
21. H. Su, K. Zheng, K. Zeng, J. Huang, and X. Zhou. STMaker: A System to Make Sense of Trajectory Data. *40th International Conference on Very Large Data Bases (VLDB)*, 7(13):1701–1704, 2014.
22. Z. Xiong, V. V. Dixit, and S. T. Waller. The development of an Ontology for driving Context Modelling and reasoning. In *IEEE 19th International Conference on Intelligent Transportation Systems (ITSC)*, pages 13–18, 2016.
23. Z. Yan, D. Chakraborty, C. Parent, S. Spaccapietra, and K. Aberer. Semantic Trajectories: Mobility Data Computation and Annotation. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 4(3):49:1–49:38, 2013.
24. L. Zhao, R. Ichise, S. Mita, and Y. Sasaki. Core Ontologies for Safe Autonomous Driving. In *14th International Semantic Web Conference, Posters and Demos Track (ISWC)*, 2015.