VSSo - A Vehicle Signal and Attribute Ontology (Short Paper)

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. 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

We propose to use semantic technologies for addressing the challenge of defining a formal model of car signals [2]. Many ontologies have been developed in order to solve problems in the automotive domain. In 2003, [10] proposed an ontology-based data access for car. [11] describes the relationship between components, failures and their symptoms. [3] 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) [13,1,9,12,8], 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[4] (SSN) ontology [5,4] is an ontology for describing sensors and their observations, 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[5] [7] (Sensor, Observation, Sample, and Actuator) for its elementary classes and properties, that was released in October 2017. Both SSN and SOSA are domain independent.

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 the Vehicle Signal Specification (VSS)[6] 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 evaluate it in Section 4. Finally we conclude and describe future work in Section 5.

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[7], which we will later use as a mean of evaluating the produced ontology.

There is a need to define a number of static properties or attributes describing either a complete vehicle or its parts. For instance:

- How many attributes does a car have?
- What is the model and release date of this car?
- What are the dimensions of this car?

[4] http://www.w3.org/ns/ssn/
[5] http://www.w3.org/ns/sosa/
[6] https://github.com/GENIVI/vehicle_signal_specification
[7] The full list of questions is available at https://github.com/klotzbenjamin/vss-ontology

- What type of transmission does this car have?
- How many doors or seats does this car have?
- On which side is located the steering wheel in this car?

A car contains numerous sensors that produce signals. We need to be able to retrieve metadata about those sensors and signals. For instance:

- Is there a signal measuring the steering wheel angle?
- Which signals are both observable and actuable?
- How many different speedometers does this car contain?
- What unit type does the signal `vss:VehicleYaw` use?
- What are the maximum values allowed for all signals from a Vehicle?
- Which signals measure a temperature and in which part are they located in the car?

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?
- Which windows are currently open?
- What is the local temperature on the driver side now?

We hypothesize that the generic modeling patterns defined in the SSN/-SOSA ontology [5] 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. 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.
- *Attributes* are the static information about a car. They are represented as leaves in the VSS tree and defined by a path starting with "Attribute" describing 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.

– *Signals* are the dynamic information about a car that is either produced by a sensor or consumed by an actuator. 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.

In its original form, VSS did not contain information about sensors or actuators. In order to describe the difference between signals measuring the same phenomenon with different sensors, we added new entries in VSS signals. We also clarified the concepts names and choices of units. Those corrections have been approved by GENIVI and are now part of evolution of this standard.

3.2 General modeling pattern

The general idea behind the design of the VSS ontology is to reuse the pattern of VSS. All branches are sub-branches of bigger branches. 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 parent branch is 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.
```

It also reuses the sets 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 and a sensor or actuator. All these entries define the specific details of an attribute or signal (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 sub-properties of a generic `vss:attribute` datatype property. All those attributes being static, there is no need to model them using a dynamic pattern. 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.TankCapacity:
        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 a dynamic modeling pattern. 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*) to model this. Observations contextualize the data with properties such as `sosa:FeatureOfInterest` (e.g. a car), the `sosa:(Simple)Result` as well as `sosa:phenomenonTime` and `geo:lat`, `geo:long` for the spatiotemporal context of the Observation. An equivalent pattern exist for actuations in SOSA.

SSN/SOSA does not define a unique unit ontology, but it is open to use multiple ones. In order to remain open, we only set restrictions on unit systems in the QUDT ontology[8][6] 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.

Concept disambiguation. VSS relies on a full path to define an attribute or a signal. 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 the base concepts. Therefore, VSS concepts are renamed for clarification. In this example, `vss:EngineSpeed` and `vss:VehicleSpeed` are actually created. Sometimes, two different paths in the VSS tree actually refer to the same concept. For example, the signal `Signal.Drivetrain.Transmission.Speed` also measures the speed of the car according to the gearbox. The class `vss:VehicleSpeed` is here unique. Its instances differ given the sensors that will produce the values and the branch hosting the sensor (Fig. 1).

[8] http://www.qudt.org

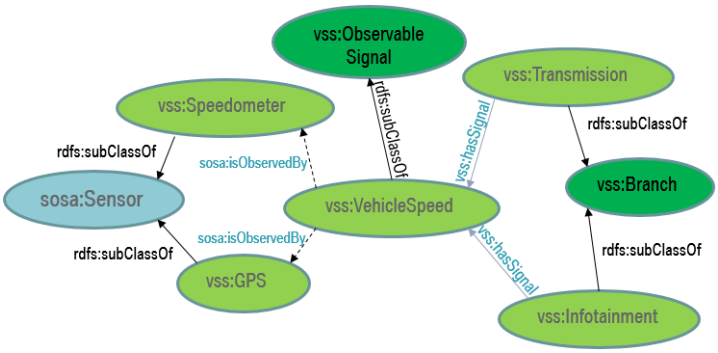

Fig. 1. Two signals representing the same concept: the vehicle speed

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. 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.

We also define a `sosa:Sensor` for all `sosa:ObservableProperty` and a `sosa:Actuator` for all `sosa:ActuatableProperty`. These sensors and actuators are 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 and consuming the data.

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.

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". 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 vss:Left` and the mirror instance a `vss:position vss:Right`.

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. We also generate synthetic traces data using the VSSo ontology. Here are simple examples of such queries:
What are the attributes of the chassis?

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

What is the current gear?

```
SELECT ?signal ?result ?time
WHERE {
  ?signal a vss:Gear.
  ?obs a sosa:Observation;
       sosa:observedProperty ?signal;
       sosa:hasSimpleResult ?result;
       sosa:phenomenonTime ?time.
}
ORDER BY DESC(?time)
LIMIT 1
```

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. In order to do so, a developer can directly use VSSo and its patterns to create new attributes, branches and signals. Another solution consists in writing the VSS extension in VSS format, and generate a new ontology. However, this second solution requires a step of validation afterwards. We extended the generator[9] with a simple health check script. 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

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

the SSN/SOSA recommendation. This new formal representation of car signals and attributes allows semantic queries. Various applications can benefit from this ontology, such as car fleet monitoring, car trajectory mining, contextual representation of a car and interaction between any car and web services.

References

1. 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.
2. 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.
3. 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.
4. A. Haller, K. Janowicz, S. J. Cox, M. Lefranois, K. Taylor, D. Le 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. *Semantic Web Journal*, 2018.
5. A. Haller, K. Janowicz, S. J. D. Cox, D. Le Phuoc, K. Taylor, and M. Lefrançois. Semantic Sensor Network Ontology. W3C and OGC Recommendation, W3C & OGC, Oct. 19 2017.
6. 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.
7. K. Janowicz, A. Haller, S. J. Cox, D. Le Phuoc, and M. Lefranois. SOSA: A lightweight ontology for sensors, observations, samples, and actuators. *Journal of Web Semantics*, 2018.
8. 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.
9. M. Madkour and A. Maach. Ontology-based context modeling for vehicle context-aware services. *Journal of Theoretical and Applied Information Technology*, 34(2), 2011.
10. 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.
11. 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.
12. 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.
13. 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.