A Car as a Semantic Web Thing: Motivation and Demonstration

Benjamin Klotz*†, Soumya Kanti Datta†, Daniel Wilms*, Raphaël Troncy† and Christian Bonnet†
*BMW Group Research, New Technologies, Innovations
Munich, Germany
Email: benjamin.bk.klotz@bmwgroup.com
†EURECOM, Sophia Antipolis, France
Email: firstname.lastname@eurecom.fr

***Abstract*—Car signal data is usually hard to access, understand and integrate for non automotive domain experts.**

In this paper, we use semantic technologies for enriching signal data in the automotive industry and access it through Web of Things interactions. This combination allows the access and integration of car data from the web.

We built VSSo, a Vehicle Signal ontology based on SOSA/SSN Observations and Actuations, and generated WoT Actions, Events and Properties, enriched with domain metadata. We mapped VSSo to a Web of Things ontology and we developed a Web of Things protocol binding with LwM2M, and made an implementation in a real car.

This implementation resulted in a first working prototype, and a number of future improvements required in order to be compliant with automotive standards.

***Keywords*-Semantic Web; Automotive; Web of Things; WoT; VSS; Ontology; LwM2M**

I. Introduction

Current and future automotive innovations are based on the interconnection of systems such as the vehicle, infrastructure back-ends and external data sources.

In current research [1], [2], [3], we experience that interconnection of various vehicle-related systems, as well as their growing autonomy requires a mean for them to understand their surroundings and share this knowledge. Vehicle produce massive amounts of data and there is room for solutions based on Web Technologies. Many standards are emerging to solve parts of this problem: GENIVI's Vehicle Signal Specification[1] defines paths and a vocabulary for car signals, SOSA/SSN [4] defines ontologies for Observations and Actuations, Sensors and Actuators, and the Web of Things defines technology and protocol-independent interactions with Web Things. All those standards enable partly a semantic enrichment of dynamic automotive data.

By combining these standards, we want to enable access and interaction to car signals, from the web, give the accessed data a meaning, make it reusable and allow semantic web developers to integrate it. The car data we are interested in is therefore the characteristics of a car and its signals.

[1]https://github.com/GENIVI/vehicle_signal_specification

In this paper, we aim to answer the research question: *How should we best define car data and link it to external knowledge through WoT interaction patterns?*.

The remaining of this paper is organized as follows. In Section II, we describe the ontological approaches to vehicle data and WoT, in Section III, we describe the design of our Vehicle Signal ontology based on GENIVI's Vehicle Signal Specification (VSS). In Section IV, we describe the pattern for integrating our Vehicle Signal ontology with a WoT ontology. Then in Section V we describe the Car Web Thing prototype we developed for the W3C WoT F2F meeting (Düsseldorf, 2017) with another contribution by defining a new Protocol Binding with LwM2M. Finally in section VI, we conclude with the future work.

II. Background

A. Ontologies for dynamic signals

Already largely used on the web, especially by search engines and the schema.org initiative, Semantic Technologies are more and more extended to physical devices in the Internet of Things[2] and automotive domain[3].

A well-though combination of ontologies may enable queries about complex driving contexts. It may include car signals, location, time and external data as well as labels tagging the driver, extracted from sensor data.

For instance in [5] we focus on two main use cases: generate segments of trajectory annotated according to the evolution of a given signal value, and a "smooth" driving percentage label attached to a trajectory when longitudinal and angular acceleration are bound.

This is made possible by combining a Vehicle Signal ontology with SOSA/SSN[4] for Observation patterns and STEP[6] for Semantic Trajectory annotation patterns.

SOSA/SSN defines terms related to signals, sensors, actuators, observations and actuations of systems. Its pattern is a base for developing a domain signal ontology.

[2]http://iot.schema.org
[3]http://auto.schema.org

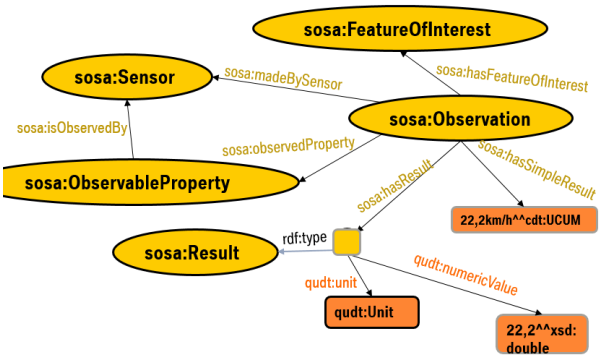

Figure 1. Overview of SOSA classes from an Observation point of view

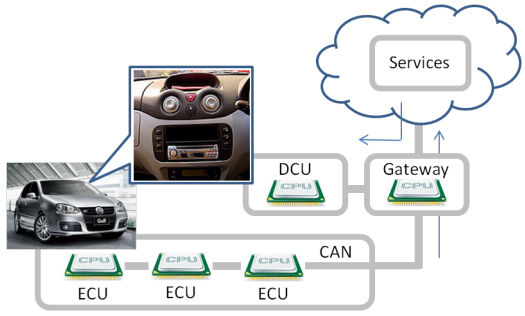

Figure 2. W3C Web of Things use case: a connected car with a cloud server

For simplification, we consider that are two types of data that can be linked to a vehicle[4]: static and dynamic. Static data are the car's *Attributes* such as its model, number of wheels, dimensions and embedded sensors. Dynamic data are car *Signals* and depend both on time and space. In addition, embedded sensors and actuators only produce dynamic data if they exist in the static car description in a list of known signals.

B. Web of Things

For interacting with heterogeneous systems following different standards in the Internet of Things (IoT), we look at the solution proposed by the Web of Things (WoT). Its goal is to allow the discovery, sharing, composition and reuse of connected physical devices in a web layer and, therefore, counter the fragmentation of the IoT[5].

The World Wide Web Consortium (W3C) has launched the Web of Things Working Group[6] (WG) in 2016. Its goal are to develop initial standards for the Web of Things, reduce the costs of development, lessen the risks to both investors and customers, and encourage exponential growth in the market for IoT devices and services.

At the heart of WoT is the WoT servient[7]: an entity consisting of a Web client, a Web server and device control capabilities. It is essentially a virtual device which provides access, controls and get statuses from physical IoT devices.

The W3C Web of Things WG presented a few use cases of servients including one about a connected car[7] as visible in Figure 2, with WoT-based services running in the back-end of the connected car. In this use case, after a discovery phase of car components through a connection gateway, the WoT servient collects data pushed from car components and allows services to access car components through its WoT interface. The collection and analysis is deployed to a fleet of cars to determine traffic patterns.

[4]www.automotive-ontology.org
[5]https://webofthings.org/2016/01/23/wot-vs-iot-12/
[6]https://www.w3.org/WoT/WG/
[7]https://w3c.github.io/wot-architecture/#connected-car

This example shows the main benefit of WoT for the automotive domain: it allows the decorrelation from automotive standard for car data - and therefore allows developers who are not automotive experts to use WoT interaction patterns with vehicles as Web Things. It also enables the collection and analysis of sensor data coming from vehicles of different models and brands. We are using WoT in our research to benefit from WoT interactions and be able to combine them in a common web layer.

In the Thing Descriptions (TD) are attached annotations about Things, capacities and interactions. The WoT ontology [8] defines those terms. In this ontology, a `wot:Thing` implements a `wot:Security`, defined as its security mechanism, and a number of `wot:InteractionPattern` that can be subclassed as `wot:Property`, `wot:Action` and `wot:Event`. Their instances are the interactions associated with a Thing, and are defined by a `wot:Link` and `wot:CommunicationProtocol` to access the device, and `wot:DataSchema` for their input/output. In addition to that wot:Property instances can have a property `wot:isMeasuredIn` to define a `om:Unit`[8].

III. VSSo MODELING

Some papers define ontologies for car sensors and controls [1], [9], or for driving context[2], [3], [10], [11] or even for ADAS[12]. They highlight the potential interest of ontological representation of car data and driving context. However, they lack an extensive representation of available car signals, sensors and actuators. Only small sets of tenth of signals are present in those ontologies and vocabularies. This limits the possibilities of ontology-based interactions with vehicle data.

This prevents people who are not domain experts from developing applications based on data outside the limited set of well-known signals that those papers define.

On another note, the W3C Automotive Working Group develops Open Web Platform specifications for exposing

[8]http://www.wurvoc.org/vocabularies/om-1.8/Unit_of_measure

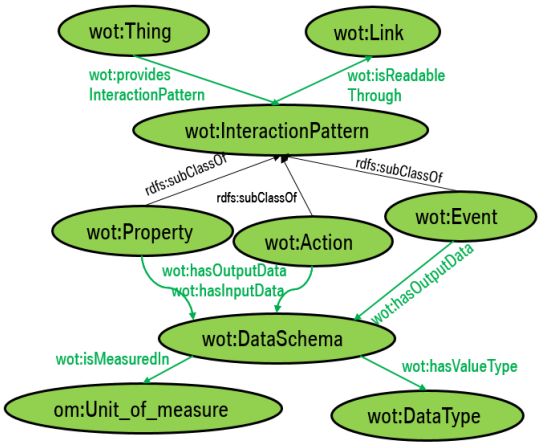

Figure 3. A set of WoT classes, as visible in a TD

vehicle signal information. One of them is the Vehicle Information Service Specification[9] (VISS): the "in-vehicle" server responsible for exposing vehicle data in a manner consistent with a given data model to enable client applications to get, set, subscribe and unsubscribe to vehicle signals and data attributes. Its current data model is the Vehicle Signal Specification (VSS) from GENIVI. The VSS is a common naming space to decouple the vehicle electrical network from its original representation to exchange data with third parties. It contains an extensive set of vehicle parts and signals, defined by a name, comment, unit and format. In its current version, the VSS defines 43 car attributes, 451 branches and 1060 signals.

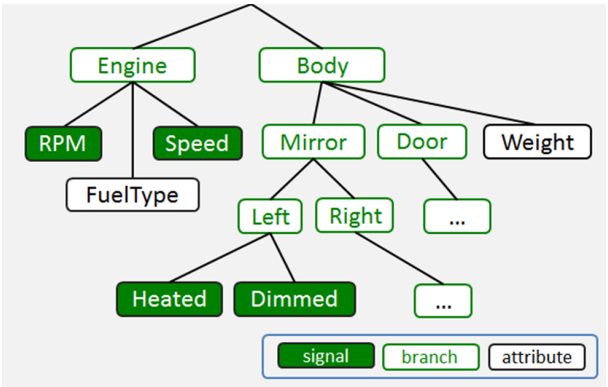

Figure 4. GENIVI Vehicle Signal Specification structure

In order to represent Observations of car signals, we adopt the SOSA/SSN[10] W3C recommendation[4]. Its pattern allows the extension with a domain ontology, which must be extensive enough. Our idea is to generate an ontology based on the VSS according to the SOSA/SSN pattern of `sosa:Observation`, `sosa:ObservableProperty` and `sosa:Sensor`.

First, we complement the VSS by adding sensor entries for signals that are produced by one. An equivalent task is done for signals associated with actuators. The remaining signals will become properties of a branch, or be linked to the class vss:VirtualSensor. For instance:

- `Signal.Drivetrain.Transmission.Speed` represent the speed sensed by the gearbox, and is measured by a speedometer.
- `Signal.Body.Mirrors.Left.Heating.Status` is the status of the actuator heater of the left mirror.
- `Signal.OBD.FuelType` is an attribute of the OBD Branch, limited in the values it can have.

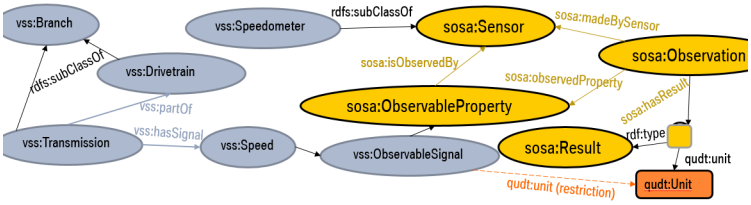

Figure 5. Modeling pattern of VSSo from SOSA, from an Observation of Speed point of view

We apply then the following pattern for creating triples. `vss:ObservableSignal` is defined as a subclass of `sosa:ObservableProperty` and `vss:ActuableSignal` a subclass of `sosa:ActuableProperty`. All signals defined in VSS are specific subclasses of them. They are defined with restrictions on their sensor/actuator and unit. All sensors are also subclasses of repectively `sosa:Sensor` and `sosa:Actuator`. Thus the object property `sosa:isObservedBy` defines the link between a signal and its sensor. Likewise, there is the property `sosa:isActedBy` between a signal and its actuator.

`vss:Branch` is defined as a component of the vehicle. All branches in VSSo are subclasses of `vss:Branch` and follow the tree structure of VSS branches using the component property `vss:partOf`.

`vss:attribute` is defined as a datatype property. Its range is a `vss:Branch`. All attributes from VSS are then defined as subProperty of `vss:attribute` with restrictions on their ranges with custom datatypes.

Because of the many exceptions, we cannot do an automatic conversion of VSS into RDF triples. Here are some modeling choices we made for consistency:

1) **Punning:** for instance `Engine.Speed`, `Drivetrain.Speed` and `Navigation.Speed` are different concepts: the first is a rotation speed, the second and third observe the same phenomenon

[9]https://www.w3.org/TR/vehicle-information-service/
[10]https://www.w3.org/TR/vocab-ssn/

but with different sensors. We define the classes `vss:RotationSpeed` and `vss:VehicleSpeed` for the 2 phenomenon.

2) **Sensors, Actuators, both or none:** some signals are only readable, consumable or not attached to any physical sensor/actuator. In the first two cases we define subclasses of `vss:ObservableSignal` or `vss:ActuableSignal`, for the second case it is a subclass of both. In the final case, we define a virtual system to be compliant with SOSA.
3) **position of branches:** some branches contain an indication about its position (e.g. "Left", "Row1"). We remove all branches defining a position and define an attribute `vss:position` on the concerned branches.
4) **structure of branches:** all branches should be part of a branch representing the whole vehicle `vss:Vehicle`. We define all branches as `vss:partOf` it.

Listing 1 is an extract from the VSS ontology describing `vss:TravelledDistance`, a signal measured by a `vss:Odometer`, with the unit `unit:Kilometer`.

Listing 1. VSSo sample: vss:VehicleSpeed signal definition in

```
vss:VehicleSpeed a rdfs:Class, owl:Class;
    rdfs:subClassOf vss:ObservableSignal;
    rdfs:label "Speed"@en;
    rdfs:comment
    "Signal.Drivetrain.Transmission.VehicleSpeed.
    Vehicle speed, as sensed by the gearbox."@en;
    rdfs:subClassOf [
        a owl:Restriction;
        owl:onProperty sosa:isObservedBy;
        owl:allValuesFrom vss:Speedometer.
    ],
    [
        a owl:Restriction;
        owl:onProperty qudt:unit;
        owl:allValuesFrom qudt:KilometerPerHour.
    ].
```

IV. VSSo-WoT Modeling Patterns

Web Things are defined in their Thing Descriptions, which are based on the WoT ontology [8]. They are usually serialised in JSON-LD using the WoT ontology as main context.

For instance a car, defined as a wot:Thing, has an interaction to control a window. It is an instance of `wot:Action`, that expects an input. It has another interaction to read its speed value. In this case, it is instantiated as a `wot:Property` and expects a `wot:DataSchema` output as well as a unit.

With VSSo, we apply the following matching rules. All `vss:ObservableSignal` and `vss:ActuableSignal` instances can be used as `wot:Property` of a Car Thing, and all `vss:ObservableSignal` instances are not writable. Likewise, all `vss:ActuableSignal` instances can be used as `wot:Action`. We do not cover the case of `wot:Event` in this research.

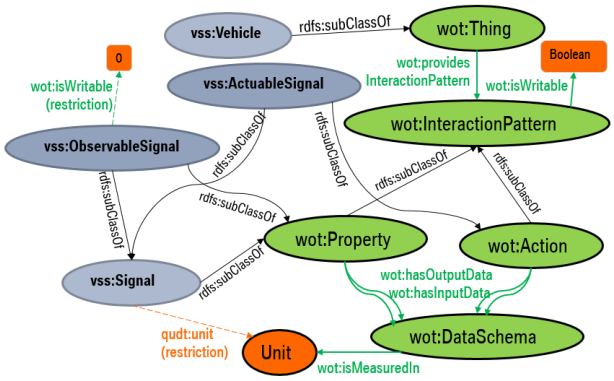

Figure 6. Modeling pattern of VSSo with WoT. VSS in blue, WoT in green, Units and literals in orange.

Listing 2. extracts from a TD representing a car and the wot:Property of the vss:VehicleSpeed

```
"@context": [
"https://w3c.github.io/wot/w3c-wot-td-context.jsonld/",
"https://w3c.github.io/wot/w3c-wot-common-context.jsonld/",
{"om":"http://www.wurvoc.org/vocabularies/om-1.8/"},
{"auto": "https://auto.schema.org/" },
{"vss": "http://automotive.eurecom.fr/vsso#" }]
"@type": ["Thing","vss:Vehicle", "auto:Car"],
"name" : "BMW 7 Series",
"auto:brand" : "BMW",
"interaction" : [
        {
          "@type": ["Property","vss:VehicleSpeed"],
          "wot:isMeasuredIn":"om:Speed_Unit",
          "name": "speed",
          "outputData": { "type": "float" },
          "observable": true,
          "writable": false,
          "link":
          [{
            "href" : "property/read/speed",
            "mediaType": "application/json"
          }]}]
```

V. Automotive Semantic Web Thing Prototype

In our prototyping, we have three challenges in regard to vehicle data-based applications:

- Find the right degree of abstraction in form of a data model for vehicle data and services,
- Transport the data to the cloud reliably, securely and efficiently
- Ease the access to both, internal and external applications.

This demonstration establishes the benefits of using a combination of ontologies and WoT patterns and was presented at the W3C WoT F2F meeting (Düsseldorf, 2017).

In this prototype, we demonstrate the feasibility of implementation of a car as a WoT servient. The prototype highlights the potential use of properties, actions and events on a motionless vehicle based on doors/windows sensors and actuators.

A. Architecture

As visible in Figure 7, the general architecture of the prototype contains 6 main parts:

1) Car data access with the computing device, through the OBD interface
2) Implementation of a LwM2M client on the computing device and server in the cloud exchanging messages over CoAP,
3) Protocol Binding: implementation of a mapping between LwM2M and WoT (Table 1)
4) Thing Description: retrieval and parsing of metadata
5) Scripting API: WoT endpoint
6) WoT client in a browser exchanging over HTTP with the WoT server.

The vehicle is connected to a computing device through its OBD dongle, which is then connected to the cloud via a LTE connection. A CoAP[11] (Constrained Application Protocol) server is running on the latter, that can notice a client running on the computing device and do GET/SET/SUBSCRIBE calls. When a sensor value is required, the client sends an OBD job to the vehicle to retrieve the raw information, then enrich it with semantic annotations based on its TD, and sends the enriched data to the server.

We use the device management protocol LwM2M[12] (LightweightM2M) specified at the Open Mobile Alliance[13] for exchanging data between the vehicle and our cloud. LwM2M is designed for remote management of sensor networks in machine-to-machine environments. It is built on CoAP and features a RESTful architectural design, with an extensible resource and data model.

A READ request can give information about a sensor value at one moment. If the server *subscribes* to speed sensor value, in fact the client will do regular READ request and the server will access it in soft real-time.

A WRITE request can update a value for an actuator. In this case, the WRITE request would also contain a parameter value pushed to the vehicle.

A discovery and a Thing Description consumption phase provides the remote servient a description of properties, actions and events that can be called through LwM2M equivalent operations on mapped objects. In this case, a mapping between LwM2M and WoT operations, as well as a definition of TD as LwM2M objects will be provided.

B. LwM2M binding (part 3 in Figure 7)

One possible implementation of LwM2M in java is the open source project Leshan[14]. Through Leshan and an additional implementation running in the vehicle, it is possible to have read and write access to selected and published data streams of the vehicle. To facilitate an easy integration of other components and domains a separate high-level, but proprietary API is implemented. An important aspect is to work on the same data model throughout the stack. Table 1 presents the WoT interactions patterns mapped between HTTP in the OEM cloud and LwM2M to reach the vehicle.

Our implementation demonstrates the usability of LwM2M as protocol for WoT, and allows its usage. This is especially relevant for applications with constrained devices.

WoT		LwM2M	Input
Interaction	Method	Method	
Property	Read	Read	Property object
	Write	Write	Property instance, parameter
Action	Invoke	Execute	Action object
	Update/cancel task	Write on instance	Action instance, parameters
Event	Subscribe	Observe	Event object
	Update/Cancel subscription	Write on instance	Event instance, parameter

Table I
WOT BINDING OF LWM2M WITH HTTP

C. Implementation notes

In our implementation, we used a Rasperry Pi as a computing device, connected to the OBD interface.

In this demonstration, we implement the following interactions:

- Properties Speed, passenger door lock,
- Actions passenger doors lock and unlock, honk
- Event speed value: built as a subscription to a property speed.

VI. CONCLUSION AND FUTURE WORK

From the experience carried out, and the feedback given at the W3C WoT F2F meeting (Düsseldorf, 2017), we see a number of improvement to the system.

Security and authorization: The automotive domain requires well-thought and secure authorization means. We did not focus on this aspect in this demonstration and this is the next requirement before being compliant with the automotive industry requirements. An existing initiative is being carried out by the W3C Automotive Working Group, which released a W3C Candidate Recommendation for Vehicle Information Service Specification [13] in February 2018. A the W3C WoT WG is experimenting different security and authentication mechanisms, it is going to be a future step for our work to make a connection between those 2 works.

Ontology Design Pattern for Web Things: We manually extended the TD with semantic metadata about the car. We use auto.schema.org for defining the car and the vss ontology for signals. We now want to generalize it for all potential signals present in the VSS, and map interactions with iot.schema.org. This will make semantic integration easier based on the common schema, and include domain description about vehicles, capacities, signals sensors and actuators.

[11] http://coap.technology/
[12] http://openmobilealliance.org/iot/lightweight-m2m-lwm2m
[13] http://openmobilealliance.org
[14] https://www.eclipse.org/leshan/

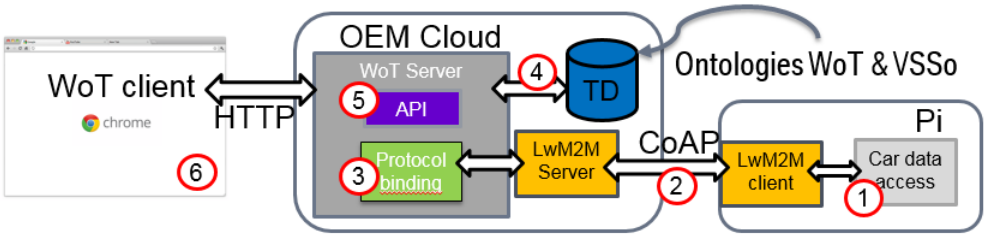

Figure 7. Prototype architecture: a WoT servient runs in the OEM clouds and interacts with a WoT client in a browser. Box colors match the color scheme of the WoT servient architecture.

Automatic generation of WoT TD: Thing Descriptions are being currently hard-coded as the specification is in a working draft. However, for the case of a car, a complete TD would contain thousands of interactions and there is a need for automatic generation of them. A future step will be to translate a rdf-based representation of a car signals, sensors and actuators into a TD with WoT interactions.

Use with cache and WoT Events: Another potential improvement for our prototype would consist in including complex event properties, using records of timed events. A use case could be to do signal analysis on a sliding window like in [5]. This would enable more complex applications based on WoT interactions and ultimately, WoT-based enrichment of car data.

Use with a context ontology: A final future step will consist in using a driving context ontology for representing abstracted information. This context could be again access through WoT interactions and allow the access and reuse of abstracted data directly from a car's back-end as a Semantic Web Thing.

References

[1] L. Zhao, R. Ichise, S. Mita, and Y. Sasaki, "Core Ontologies for Safe Autonomous Driving," in *14^{th} International Semantic Web Conference, Posters and Demos Track (ISWC)*, 2015.

[2] A. Armand, D. Filliat, and J. Ibaez-Guzman, "Ontology-based context awareness for driving assistance systems," in *IEEE Intelligent Vehicles Symposium Proceedings*, 2014, pp. 227–233.

[3] M. Madkour and A. Maach, "Ontology-based context modeling for vehicle context-aware services," vol. 31, 2011.

[4] K. Janowicz, S. Cox, K. Taylor, D. L. Phuoc, M. Lefrançois, and A. Haller, "Semantic sensor network ontology," W3C, W3C Recommendation, Oct. 2017, https://www.w3.org/TR/2017/REC-vocab-ssn-20171019/.

[5] B. Klotz, R. Troncy, D. Wilms, and C. Bonnet, " Generating Semantic Trajectories Using a Car Signal Ontology," April 2018, to appear.

[6] T. Paiva Nogueira, "A Framework for Automatic Annotation of Semantic Trajectories," Ph.D. dissertation, Université Grenoble Alpes, 2017.

[7] K. Kajimoto, R. Matsukura, J. Hund, M. Kovatsch, and K. Nimura, "Web of things (wot) architecture," W3C, W3C Unofficial Draft, Feb. 2018, https://w3c.github.io/wotwg/architecture/wot-architecture.html.

[8] V. Charpenay, S. Käbisch, and H. Kosch, "Introducing Thing Descriptions and Interactions: An Ontology for the Web of Things."

[9] M. Feld and C. Müller, "The Automotive Ontology: Managing Knowledge Inside the Vehicle and Sharing it Between Cars," in *3^{rd} International Conference on Automotive User Interfaces and Interactive Vehicular Applications*, Salzburg, Austria, 2011, pp. 79–86.

[10] 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)*, 2016, pp. 13–18.

[11] S. Fuchs, S. Rass, and K. Kyamakya, "Integration of Ontological Scene Representation and Logic-Based Reasoning for Context-Aware Driver Assistance Systems," *ECEASST*, vol. 11, 2008. [Online]. Available: http://journal.ub.tu-berlin.de/index.php/eceasst/article/view/127

[12] 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)*, vol. 1, no. 3, July 2010.

[13] P. Kinney, A. Crofts, W. Lee, and K. Gavigan, "Vehicle information service specification," W3C, Candidate Recommendation, Feb. 2018, https://www.w3.org/TR/2018/CR-vehicle-information-service-20180213/.