A Knowledge Layer in Data-Centric Architectures in the Automotive Industry

Haonan Qiu, Adel Ayara and Christian Muehlbauer
BMW Group, Germany

Keywords: Knowledge Representation, Reasoning, Data-Centric Architecture, Automotive Industry.

Abstract: As the automotive industry continues its evolution towards connected and autonomous vehicles, the need to adopt a data-centric architecture for providing sophisticated driving functions increasing important. Compared to traditional application-centric architecture, data-enteric architecture allows the collection, aggregation and analysis of data from various sources. However, such architecture lack of inference capability makes it not leverage the full potential of established AI technologies such as knowledge representation and reasoning. In this work, it is our goal to investigate how we can incorporate a knowledge layer in a data centric architecture that allows reasoning and thus support decision making. A prototype is implemented and deployed in both vehicle and backend, and the feasibility is evaluated through a real-world experiment.

1 INTRODUCTION

In 1977, the first production car that deployed a small piece of software was the General Motors Oldsmobile Toronado which had an Electronic Control Unit (ECU) that managed electronic spark timing Bereisa (1983). The first software-based solutions were purely local, functionally and technically isolated, and had no connections between them. These independent and unconnected pieces of software used to run on a single dedicated ECU. Applications were created in machine code directly, with a minimal amount of abstraction used.

Therefore, ECUs designed for specific tasks, along with their sensors and actuators, formed the basic architecture of vehicles. In the early 1990s, ECUs started to communicate with each other and were thus able to exchange data. As a consequence, the automotive industry started to introduce functions distributed over several ECUs, connected by bus systems as the underlying communication infrastructure. Software coupled with ECUs is increasing as the need arises. In less than 50 years, the amount of software has evolved from zero to hundreds of millions of lines of code [1].

Besides the software complexity, the current vehicle architecture results in isolated data processing, and the value gained from data during execution remains within the application. This architectural approach is sometimes referred to as an *application-centric* architecture, wherein each application uses its own schema to modify and transmit data based on its specific requirements. However, this design choice results in the process logic being tightly coupled with the application, leading to challenges in long-term maintenance (McComb, 2018). Additionally, modern vehicles generate vast amounts of data from various sources, such as sensors and cameras. According to the the calculation reported in (Wright, 2023), the sensors in an autonomous vehicle record between 1.4 terabytes (TB) and around 19 TB per hour. This big volume of heterogeneous data presents yet another challenge to an application-centric approach's ability to effectively process and extract valuable insights.

To address the aforementioned issue, there is a shift from application-centric approaches to data-centric approaches (Alvarez-Coello et al., 2021). It focuses on improving the vehicle data architecture, which plays a crucial role in supporting the advancement of the automotive industry in the upcoming decade, particularly in terms of connected vehicles, autonomous systems, and shared mobility. With a focus on a data-centric perspective, the authors of (Alvarez-Coello et al., 2021) outline a data architecture based on the well-known Data, Information, Knowledge, Wisdom (DIKW) hierarchy.

This paper is a continuation of work done in (Alvarez-Coello et al., 2021), aiming to propose a knowledge layer within the data-centric architecture

[1] https://spectrum.ieee.org/software-eating-car

Qiu, H., Ayara, A. and Muehlbauer, C.
A Knowledge Layer in Data-Centric Architectures in the Automotive Industry.
DOI: 10.5220/0012255000003598
In *Proceedings of the 15th International Joint Conference on Knowledge Discovery, Knowledge Engineering and Knowledge Management (IC3K 2023) - Volume 2: KEOD*, pages 296-303
ISBN: 978-989-758-671-2; ISSN: 2184-3228

that applies Semantic Web technologies. The main contributions are as follows:

– We propose a knowledge layer based on OWL 2 RL ontologies (Hitzler et al., 2009), Datalog rules (Abiteboul et al., 1995), and SPARQL queries (W3C SPARQL Working Group, 2013), to deal with both knowledge abstraction and decision-making.

– We propose a generic approach for exchanging messages between the information layer and the knowledge layer.

– We present an application scenario with designed ontologies and rules for the proposed architecture.

– We evaluate the proposed architecture and deployed it in both the car and the backend with a real-world experiment.

The remainder of this paper is organized as follows: Section 2 presents related works. In Section 3, we present the proposed architecture, and Section 4 describes the application scenario. In Section 5, we evaluate the approach with a real-world experiment. Section 6 concludes the paper.

2 RELATED WORK

In this section, we provide existing works about three areas whose intersection this work resides in: data-centric approaches, ontologies, and reasoning in automotive industry.

Data-centric. The key characteristic of a data-centric architecture is that the data are the central asset and their storage and governance are the first steps of the process, which precede the creation of any application or service. Nakamoto et al. (2014) propose an IoT-based data-centric software architecture using a data stream management system for all sensors, so that downstream application can get relevant data as required. Kugele et al. (2018) propose a data-centric communication based on publish-subscribe mechanism, focusing on data rather than actions. They also propose to use containerization techniques for portability and continuous integration and delivery. Yun et al. (2017) propose an architect including data-centric middleware for the digital twin platform. None of the aforementioned studies addressed the integration of a knowledge layer providing knowledge abstraction and reasoning capability on top of data. Nevertheless, their research efforts have laid the groundwork for this paper.

Ontologies. Ontologies have been applied in theautonomous industry and related fields. Based on the VSS and the Sensor, Observation, Sample and Actuator (SOSA) ontology(Janowicz et al., 2019), Klotz et al. (2018a) create an ontology for vehicle signals and sensors named VSS ontology (VSSo) (Klotz et al., 2018b). It is used in combination with the STEP ontology (Nogueira et al., 2018), an extension of a geo-ontology in (Hu et al., 2013), to map car trajectories using the ontology's vocabulary and label the segments as “smooth” or “not smooth” based acceleration range. Alvarez-Coello and Gómez (2021) use an ontology-based on approach to combine vehicle-related data and label the application-specific data with semantics such that queries that are stable over time, application-specific data can be reused, and more semantics are possible. The limitation of the described approaches is that reasoning is not incorporated.

Reasoning. Work aimed at enabling vehicles' situation awareness and decision-makings exits in the literature. Armand et al. (2014) propose reasoning using rules for providing context awareness to driving assistant systems so that the vehicle can perform human-like reasoning about its surroundings using map data and sensor data. Buechel et al. (2017) focus on the traffic scene modelling with decision-making rules derived from traffic regulations. Similarly, the approach proposed by Zhao et al. (2017) also uses rules to encode traffic rules, with the difference that the sensor data is treated as RDF streams. However, vehicle signal data is not included in any of the studies.

3 APPROACH

In this section, we describe the proposed overall architecture, followed by a detailed explanation of the knowledge layer.

3.1 Architecture

Figure 1Conceptual architecture. shows the proposed architecture in accordance with the DIKW hierarchy (Rowley, 2007). In this architecture, content flows generally from bottom to top, and the higher the data is represented in the hierarchy, the more explicit context will be immediately accessible for future use. The **data layer** represents the raw data from different sources, such as vehicle boardnet signals, data in the cloud, or data on mobile devices.

The **information layer** comprises organized data following the defined schemas, like Vehicle Signal Specification (VSS) (Klotz et al., 2018b) and Person Data Model (PDM). The PDM contains information about drivers and passengers in a variety of driving

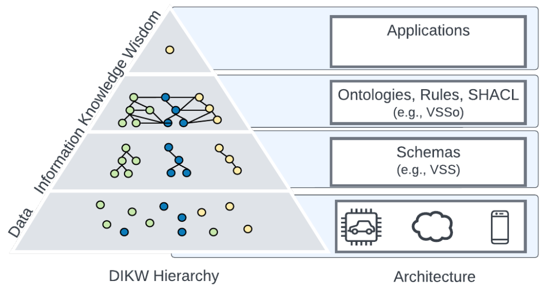


Figure 1: Conceptual architecture.

situations. The schemas within this layer take the form of a tree-structured taxonomy, serving as direct interfaces for retrieving structured data. In this layer, *data middleware* technologies can be used to transform unstructured data streams into structured data based on a suitable data schema and to synchronize data across multiple components.

The **knowledge layer** contains ontologies from different domains and makes it possible to integrate heterogeneous sources such as the Vehicle Signal Specification ontology (Vsso) and the Person Data Model ontology (PDMo). Together with background ontologies, the rules in this layer serve as input for the reasoner to perform tasks such as semantic enrichment, knowledge abstraction, and context reasoning for decision-making. Shapes Constraint Language (SHACL) files not only define constraints that validate and shape the structure of data (Knublauch and Kontokostas, 2020), but also serve as a bridge between the schema used in the information layer and the terminologies of the ontology used in the knowledge layer (Taelman et al., 2019).

The **wisdom layer** refers to the phase in which knowledge derived from the previous layer is used for making strategic decisions and generating actionable knowledge. From this point, it is possible to develop use-case-specific applications that query the insights through the knowledge layer and execute the corresponding actions.

3.2 Knowledge Layer

The knowledge layer has two primary responsibilities: data conversion and reasoning. We encapsulate this task into a component named *Inference Engine* as shown in Figure 2Inference engine..

3.2.1 Data Conversion

Data conversion is the process of transforming data that is formatted according to a tree-structured schema in the information layer into RDF graph

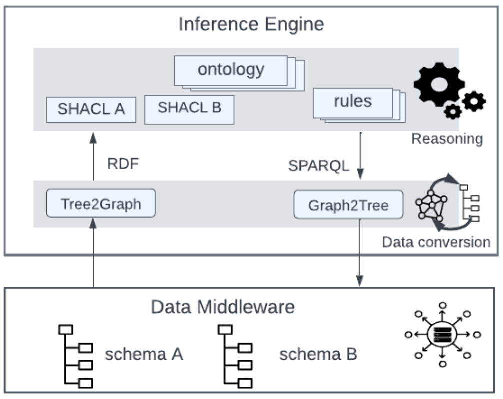


Figure 2: Inference engine.

```
val:Person a sh:NodeShape;
 sch:name "Person";
 sh:targetClass ont:Person;
 sh:property [
  sch:name "privateAddress";
  sh:node val:PrivateAddress;
  sh:path ont:hasPrivateAddress;
  sh:class ont:PrivateAddress;
  sh:maxCount 1;
 ].
```

Listing 1: An example of SHACL definition.

data that is defined by classes and properties from the ontology in the knowledge layer, and vice versa. The SHACL files define the correspondence between the shape, schema and the ontology, thereby serving as a bridge between the information layer and the knowledge layer. Specially, sch:name[2] maps the SHACL shapes to the elements in the tree-structured schema, sh:targetClass[3] or sh:class maps the SHACL shapes to the ontology classes, and sh:path maps to the ontology object property or data property. The ontology entities are prefixed with ont[4] (see Listing 1).

Tree2Graph. The values of "sch:name:" are mapped to the elements in the schema name. For example, Person, privateAddress, City are mapped to the elements in the name Person.privateAddress.City from a message shown in Listing 2. With this mapping, the corresponding ontology classes and properties are retrieved using a SPARQL query with the *input1* and *input2* replaced by the two consecutive elements from the message's name (see Listing 3). The SPARQL

[2]prefix sch: <http://groupontology.bmwgroup.net/bmw-sch#>

[3]prefix sh: <http://www.w3.org/ns/shacl#>

[4]prefix ont: <http://groupontology.bmwgroup.net/bmw-ont# >

query is executed $n-1$ times if n is the number of the elements in the message name and $n > 1$. The RDF triples then can be generated using the query result including the class, object property and data property and the value from the original message.

```
{
  "name": "Person.privateAddress.City",
  "value": "Munich"
}
```

Listing 2: The message recived from the information layer.

```
SELECT ?c1 ?op ?dp ?c2
WHERE {?S sch:name input1;
sh:targetClass ?c1;
{?S sh:property [
sch:name input2; sh:path ?op; sh:class
    ?c2].
}UNION{?S sh:property [
sch:name input2;
sh:path ?dp; sh:datatype ?dt].}
}
```

Listing 3: SPARQL query for retrieving ontology classes and properties.

Graph2Tree. The conclusion drawn by the reasoner must be synchronized with the other components and cloud storage. This is accomplished by returning the inference to the information layer, where the data middleware resides. By retrieving inferences through SPARQL queries as variable names are formatted based on the schema, the query result can be converted to messages whose name is the query variable and whose value is the query variable's value. Listing 4 shows the generated message based on the query result. The message name Person.jobAddress.Street is from the query variable name. The conversion is in generic form, making the method independent of particular data messages.

```
{
  "name":"Person.jobAddress.Street",
  "value":"Hardtweg"
}
```

Listing 4: The message sent to information layer.

3.2.2 Reasoning

Reasoning plays a central role in our architecture. In the next section, an application scenario is used to illustrate reasoning abilities such as knowledge abstraction and context reasoning.

4 APPLICATION

In this section, we describe the application scenario shown in Figure 3Application scenario.. Drivers typically travel between their home and their place of employment or another frequently visited location on a regular basis. A daily driving pattern can be inferred from past trips. Together with the commuter's private address and traffic information in the backend, the commuter will receive a business address suggestion and traffic congestion notification.

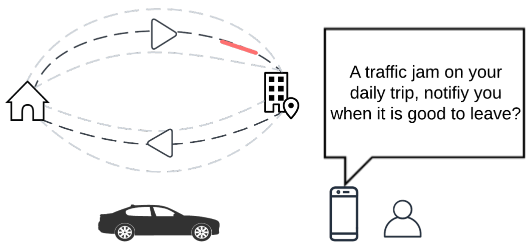

A traffic jam on your daily trip, notifiy you when it is good to leave?

Figure 3: Application scenario.

4.1 Ontologies

Figure 4Core concepts of the ontological model. shows the overview of the designed ontology with modules: the Daily User Trip ontology (DUTo), the Person Data Model ontology (PDMo), the Vehicle Signal Ontology (VSSo) and the High-level Map ontology (HLMo).

- *DUTo* represents the daily user trip of the vehicle, reusing partial content from the Geo-ontology design pattern (Hu et al., 2013). DailyUserTrip is the main concept that describes the path on which a vehicle travels over time. It may have more than one Segment if the vehicle stopped at multiple locations during the journey. A Segment has a starting Fix and an ending Fix. A Fix is defined as a point $\{latitude, longitude\}$ with a timestamp indicating the position of a moving object at an instant of time. A TrafficJam may happen on some RoadParts within a period of time, thus affecting the corresponding DailyUserTrip.
- *VSSo* is used to represent vehicle signals (Klotz et al., 2018b). We reused CurrentLocation and DrivingSatus.CurrentLocation describes the received GPS position at each timestamp. The DrivingStatus indicates the mode of vehicle motion, such as standby, driving, or engine-off.
- *PDMo* describes the person in the context of the driving scenario, such as the driver or passenger. A Person owns Vehicles, and can have a BusinessAddress and a PrivateAddress. The location of the

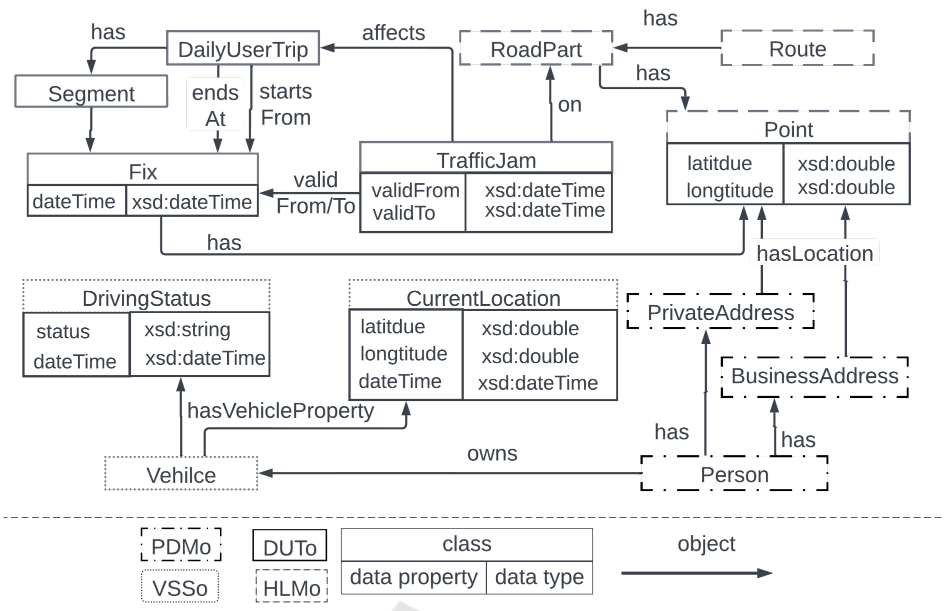


Figure 4: Core concepts of the ontological model.

address is described by a Point.

- *HDLM* describes the high-level map knowledge (Qiu et al., 2020). We reused the concepts of Road, Route and Point to represent navigational information. A Route consists of some RoadPart, and the geometry of each RoadPart is described by some Point with coordinates.

4.2 Rules

The Datalog rules are used to infer daily user trips, suggest user business address, and decide whether the traffic jams affect the daily user trips. In detail, the daily user trips are derived from vehicle signal data streams through two steps: 1) abstraction and 2) aggregation (see Figure 5Inference flow for the daily user trips.).

1) Abstraction. Vehicle signal data streams represented by VSSo instances are incrementally abstracted to the instances of Fix, Segment and DailyUserTrip in DUTo respectively. We further divide Fix into StartFix and EndFix, two subclasses representing the start and end fixes of a trip. If the DrivingStatus is "Ready_To_Drive", the instance of StartFix are derived as follows:

StartFix(f), dateTime(f,d), hasLocation(f,p),
Point(p), latitude(p,m), longitude(p,n) $\leftarrow$
    Vehilce(v), hasVehilceProperty(v,c),
    CurrentLocation(c), dateTime(c,d),
    latitude(c,m), longitude(c,n)
    hasVehilceProperty(v,ds), DrivingStatus(ds),

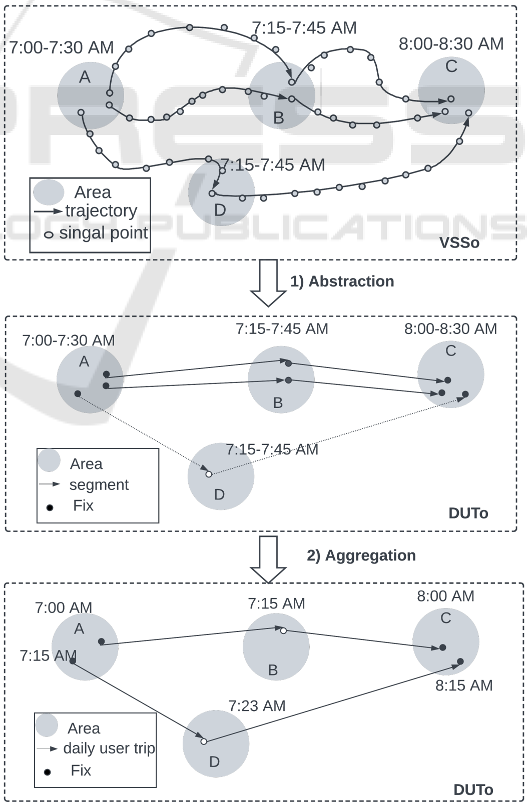


Figure 5: Inference flow for the daily user trips.

statue$(ds, "Ready_To_Drive")$, dateTime(ds,d),
BIND(SKOLEM("f", d) as f),

$\mathtt{BIND(SKOLEM("}p\mathtt{"},\ d)\ \mathtt{as}\ p).$

As vehicle signal data streams arrive, the instances of Segment are inferred incrementally based on the corresponding StartFix instance and the EndFix instance.:

$$
\begin{aligned}
&\mathsf{Segment}(s), \mathsf{startsFrom}(s, f_3),\\
&\quad \mathsf{endsAt}(s, f_2) \leftarrow\\
&\quad \mathsf{EndFix}(f_2),\\
&\quad \mathtt{AGGREGATE}(\mathsf{dateTime}(f_2, d_2),\\
&\quad \mathsf{StartFix}(f_1), \mathsf{dateTime}(f_1, d_1),\\
&\quad \mathsf{sameDate}(d_2, d_1), \mathtt{FITLER}(d_2 > d_1)\\
&\quad \mathtt{ON}\ f_2\ \mathtt{BIND}\ \mathtt{MAX}(d_1)\ \mathtt{AS}\ d_{max}),\\
&\quad \mathsf{hasLocation}(f_2), p_2), \mathsf{dateTime}(f_2), d_3),\\
&\quad \mathsf{StartFix}(f_3), \mathsf{hasLocation}(f_3), p_3),\\
&\quad \mathsf{dateTime}(f_3), d_{max}), \mathsf{sameDate}(d_3, d_{max}),\\
&\quad \mathtt{NOT}\ \mathsf{sameArea}(p_2, p_3),\\
&\quad \mathtt{BIND(SKOLEM(}\text{"}s\text{"},\ d_{max},\ d_3)).
\end{aligned}
$$

2) Aggregation. The similar segments are aggregated into a representative segment, which will be part of the inferred daily user trip. If the StartFix and EndFix of two Segment are located in the same area, and the starting time and ending time fall within 30-minute range, then the relationship isSimilarTo is derived as follows:

$$
\begin{aligned}
&\mathsf{isSimilarTo}(m_1, m_2) \leftarrow\\
&\quad \mathsf{Segment}(m_1),\ \mathsf{startsFrom}(m_1, o_1),\\
&\quad \mathsf{dateTime}(o_1, s_1)\ \mathsf{hasLocation}(o_1, t_1),\\
&\quad \mathsf{endsAt}(m_1, p_1),\ \mathsf{dateTime}(p_1, e_1),\\
&\quad \mathsf{hasLocation}(p_1, u_1), \mathsf{Segment}(m_2),\\
&\quad \mathsf{startsFrom}(m_2, o_2), \mathsf{dateTime}(o_2, s_2),\\
&\quad \mathsf{hasLocation}(o_2, t_2), \mathsf{endsAt}(m_2, p_2),\\
&\quad \mathsf{dateTime}(p_2, e_2), \mathsf{hasLocation}(p_2, u_2),\\
&\quad \mathsf{sameArea}(t_1, t_2), \mathsf{sameArea}(u_1, u_2)\\
&\quad \mathtt{NOT}\ \mathsf{sameDate}(s_1, s_2),\\
&\quad \mathtt{FILTER}(\mathtt{ABS}(s_1 - s_2) \leq 30\ \mathtt{\&\&}\ \mathtt{ABS}(e_1 - e_2) \leq 30).
\end{aligned}
$$

The isSimilarTo has properties of symmetry, reflexivity, and transitivity to ensure all inferences about this relationship among Segments are made. The daily user trip can be derived by aggregating the segments that are in isSimilarTo relationship. We define the daily user as the trip comprising segments with the earliest start among all segments that are related by insSimilarTo.

5 IMPLEMENTATION AND EVALUATION

This section presents the implementation of the proposed architecture and the evaluation of the application scenario through a real-world experiment.

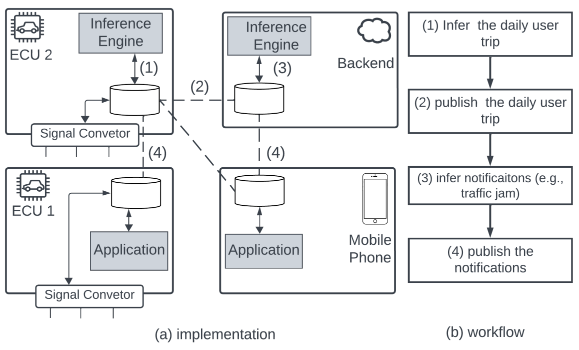


Figure 6: Overview of implementation.

5.1 Implementation

We have implemented the inference engine in C++ using RDFox 6.0 APIs[5] for importing dynamic data, reasoning and querying the inference result. The inference engine is deployed both in the vehicle and on the cloud along with a data middleware solution provided by MongoDB[6] as shown in Figure 6Overview of implementation.a. The inference result can be visualized through the font-end application deployed on the vehicle and the mobile phone. The workflow with the main steps are shown in Figure 6Overview of implementation.b. The Nominatim search API[7] is used to search OpenStreetMap (OSM) data for coordinates by address and addresses for a given coordinate.

5.2 Real-World Experiment

To evaluate whether our approach works in real-time, we conducted tests with the intelligent vehicle BMW iX on the route shown in Figure 7Experiment area.. The vehicle is equipped with a computer on which the data middleware and the implemented inference engine are deployed. Four experimental journeys have been conducted between A (BMW Research and Technology House) and B (Hardtweg 22, 85748 Garching).

5.2.1 Data

In-vehicle. Table 1An example of transmitted sensor data in a trip. Status {5=STANDBY, 8=READY_TO_DRIVE, 10=DRIVING, 12=DRIVING_END}. shows the vehicle signal data for one trip, which is transmitted in real-time to the data middleware, and later published to the inference engine. At

[5]https://docs.oxfordsemantic.tech/6.0/welcome-to-rdfox.html

[6]https://www.mongodb.com/

[7]https://nominatim.org/release-docs/latest/api/Search/

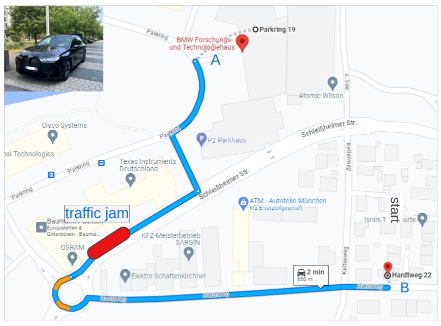


Figure 7: Experiment area.

Table 1: An example of transmitted sensor data in a trip. Status {5=STANDBY, 8=READY_TO_DRIVE, 10=DRIVING, 12=DRIVING_END}.

timestamp	latitude	longitude	status
1693320978	48.251581	11.637744	5
1693321169	48.251581	11.637744	8
1692774450	48.251581	11.637744	10
...			
1692774648	48.249755	11.63980	12

each timestamp, the sensor data transmitter sends data of latitude, longitude, and car driving status. The vehicle signal data is converted into RDF data via Tree2Graph using the VSSo ontology.

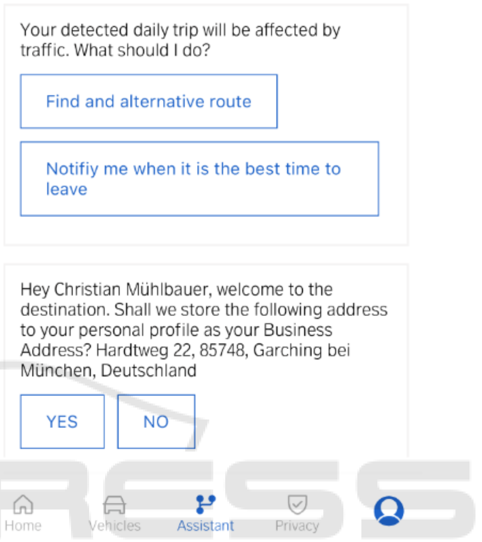
Backend. We configured the private address of the driver as A's address and simulated the traffic jam information with valid duration and location specified on the path of the experimental area (see Figure 7Experiment area.).

5.2.2 Result

Table 2Inferred daily user trips from the vehicle. shows the inferred two daily user trips in total. Each time a new daily user trip is derived, a SPARQL query is executed to retrieve the corresponding result, which is then transmitted back to the data middleware. After the daily user trip is transmitted to the backend inference engine, it is able to identify the occurrence of a traffic jam event. Subsequently, a notification is generated and transmitted to the user's mobile phone application, where it is displayed. Also, the business address is deduced and the user is prompted to save it in the system to his personal profile (see Figure 8Notifications from the mobile phone.).

Table 2: Inferred daily user trips from the vehicle.

	DailyUserTrip	
	(1)	(2)
startTime	14:59:30	15:02:52
startLocation	48.251581	48.249582
	11.637744	48.249582
endTime	15:02:45Z	15:04:08Z
endLocation	48.2495822	48.2515647
	11.639152	11.637710


Your detected daily trip will be affected by traffic. What should I do?

Hey Christian Mühlbauer, welcome to the destination. Shall we store the following address to your personal profile as your Business Address? Hardtweg 22, 85748, Garching bei München, Deutschland

Find and alternative route
Notify me when it is the best time to leave
YES
NO
Home
Vehicles
Assistant
Privacy

Figure 8: Notifications from the mobile phone.

6 CONCLUSION AND FUTURE WORK

In this paper, we propose a knowledge layer that is designed for data-centric architecture in the automotive industry using Semantic Web technologies. The technologies enable the integration of diverse domain ontologies, facilitate knowledge abstraction, and provide reasoning capabilities to support decision-making processes. We presented the mechanism to perform generic data conversion between the information layer and the knowledge layer. We explain an application scenario that utilizes the proposed architecture and present the designed ontology and rules. The rules can be reused because the underlying ontology is adapted from well-defined ontologies. Furthermore, we have developed a prototype using APIs from RDFox. The prototype was successfully deployed in both the vehicle and the backend. The proposed approach was then evaluated through a real-world experiment. The results demonstrate the feasibility of adopting a knowledge layer using Semantic Web technologies for abstracting knowledge from vehicle data streams and providing insights for decision-making.

Furthermore, the results indicate that the application within the knowledge layer can be placed in either the vehicle or the backend, leaving computational resources and data transmission costs as the primary factors impacting the decision of its deployment site.

In future work, we intend to evaluate the approach with more use cases by adding only ontologies and rules to show the scalability of the proposed architecture and to look into best practises. Additionally, we plan to extend the knowledge layer by including machine learning techniques for obtaining further insights from vehicle-related data and investigating how machine learning and reasoning can work together.

REFERENCES

Abiteboul, S., Hull, R., and Vianu, V. (1995). *Foundations of databases*, volume 8.

Alvarez-Coello, D. and Gómez, J. M. (2021). Ontology-based integration of vehicle-related data. In *2021 IEEE 15th International Conference on Semantic Computing (ICSC)*, pages 437–442.

Alvarez-Coello, D., Wilms, D., Bekan, A., and Gómez, J. M. (2021). Towards a data-centric architecture in the automotive industry. *Procedia Computer Science*, 181:658–663.

Armand, A., Filliat, D., and Ibañez-Guzman, J. (2014). Ontology-based context awareness for driving assistance systems. In *2014 IEEE intelligent vehicles symposium proceedings*, pages 227–233. IEEE.

Bereisa, J. (1983). Applications of microcomputers in automotive electronics. *IEEE Transactions on Industrial Electronics*, IE-30:87–96.

Buechel, M., Hinz, G., Ruehl, F., Schroth, H., Gyoeri, C., and Knoll, A. (2017). Ontology-based traffic scene modeling, traffic regulations dependent situational awareness and decision-making for automated vehicles. In *2017 IEEE Intelligent Vehicles Symposium (IV)*, pages 1471–1476. IEEE.

Hitzler, P., Krötzsch, M., Parsia, B., Patel-Schneider, P. F., and Rudolph, S. (27 Oct 2009). OWL 2 Web Ontology Language: Primer (2nd Edition).

Hu, Y., Janowicz, K., Carral, D., Scheider, S., Kuhn, W., Berg-Cross, G., Hitzler, P., Dean, M., and Kolas, D. (2013). A geo-ontology design pattern for semantic trajectories. In *Spatial Information Theory: 11th International Conference, COSIT 2013, Scarborough, UK, September 2-6, 2013. Proceedings 11*, pages 438–456. Springer.

Janowicz, K., Haller, A., Cox, S. J., Le Phuoc, D., and Lefrançois, M. (2019). Sosa: A lightweight ontology for sensors, observations, samples, and actuators. *Journal of Web Semantics*, 56:1–10.

Klotz, B., Troncy, R., Wilms, D., and Bonnet, C. (2018a). Generating semantic trajectories using a car signal ontology. In *Companion Proceedings of the The Web Conference 2018*, pages 135–138.

Klotz, B., Troncy, R., Wilms, D., and Bonnet, C. (2018b). Vsso: The vehicle signal and attribute ontology. In *SSN@ ISWC*, pages 56–63.

Knublauch, H. and Kontokostas, D. (2020). Shapes constraint language (shacl). (Accessed on 08/08/2023).

Kugele, S., Hettler, D., and Peter, J. (2018). Data-centric communication and containerization for future automotive software architectures. In *2018 IEEE International Conference on Software Architecture (ICSA)*, pages 65–6509. IEEE.

McComb, D. (2018). *SOFTWARE WASTELAND: how the application-centric mindset is hobbling our enterprises*. Technics Publications.

Nakamoto, Y., Yamaguchi, A., Sato, K., Honda, S., and Takada, H. (2014). Toward data-centric software architecture for automotive systems-embedded data stream processing approach. In *2014 IEEE 11th Intl Conf on Ubiquitous Intelligence and Computing and 2014 IEEE 11th Intl Conf on Autonomic and Trusted Computing and 2014 IEEE 14th Intl Conf on Scalable Computing and Communications and Its Associated Workshops*, pages 586–589. IEEE.

Nogueira, T. P., Braga, R. B., de Oliveira, C. T., and Martin, H. (2018). Framestep: A framework for annotating semantic trajectories based on episodes. *Expert Systems with Applications*, 92:533–545.

Qiu, H., Ayara, A., and Glimm, B. (2020). Ontology-based processing of dynamic maps in automated driving. In *KEOD*, pages 98–107.

Rowley, J. (2007). The wisdom hierarchy: representations of the dikw hierarchy. *Journal of information science*, 33(2):163–180.

Taelman, R., Vander Sande, M., and Verborgh, R. (2019). Bridges between graphql and rdf. In *W3C Workshop on Web Standardization for Graph Data. W3C*.

W3C SPARQL Working Group (21 Mar 2013). SPARQL 1.1 Overview.

Wright, S. (accessed on 2 July 2023). Autonomous cars generate more than 300 TB of data per year.

Yun, S., Park, J.-H., and Kim, W.-T. (2017). Data-centric middleware based digital twin platform for dependable cyber-physical systems. In *2017 ninth international conference on ubiquitous and future networks (ICUFN)*, pages 922–926. IEEE.

Zhao, L., Ichise, R., Liu, Z., Mita, S., and Sasaki, Y. (2017). Ontology-based driving decision making: A feasibility study at uncontrolled intersections. *IEICE Transactions on Information and Systems*, E100.D(7):1425–1439.